INFORMATION AND COMPUTATION **129**, 107–141 (1996)
ARTICLE NO. 0077

CPO Models for Compact GSOS Languages

Luca Aceto* and Anna Ingólfsdóttir†

Basic Research in Computer Science‡, Department of Computer Science, Aalborg University, Fredrik Bajersvej 7E, 9220 Aalborg Ø, Denmark
E-mail: {luca,annai}@iesd.auc.dk

In this paper, we present a general way of giving denotational semantics to a class of languages equipped with an operational semantics that fits the GSOS format of Bloom, Istrail, and Meyer. The canonical model used for this purpose will be Abramsky's domain of synchronization trees, and the denotational semantics automatically generated by our methods will be guaranteed to be fully abstract with respect to the finitely observable part of the bisimulation preorder. In the process of establishing the full abstraction result, we also obtain several general results on the bisimulation preorder (including a complete axiomatization for it), and give a novel operational interpretation of GSOS languages.

1. INTRODUCTION

This study is part of an on-going research programme on the meta-theory of process description languages. This line of research aims to contribute to the systematic development of process theory by offering results that hold for classes of process description languages. As these languages are often equipped with a Plotkin-style structural operational Semantics (SOS) [72], this way of giving semantics to processes has been a natural handle to establish results that hold for all languages whose semantics is given by means of inference rules that fit a certain format. Examples of the kind of meta-theoretic results that have been systematically derived from the form of the SOS rules may be found in, e.g., [6, 7, 14, 17, 20, 24, 25, 36, 37, 54, 80, 81, 89, 90, 92, 93]. So far, this line of research has produced a wealth of results which generalize and explain several of the most important theorems and constructions in process theory. For example, given a language with an SOS semantics, an examination of the SOS rules is often all that is needed to guarantee that a notion of behavioral equivalence or preorder will be preserved by the constructs in the language [17, 22–24, 37, 89, 90], that a process equivalence can be equationally characterized [5, 7], or that a language is implementable [24, 91].

Following a bias towards operational methods in process theory that dates back to Milner's original development of the theory of CCS [60, 66], most of the work reported in the aforementioned references is concerned with operational, axiomatic semantics[1] for processes and the relationships between the two. In particular, it is by now clear that it is often possible to automatically translate an operational theory of processes into an axiomatic one [7]. Moreover, in certain circumstances, it is also possible to derive an SOS semantics from an axiomatic one, as witnessed by the developments in [7, 49].

Axiomatic semantics and proof systems for programming and specification languages are often closely related to denotational semantics for them, particularly if the Scott–Strachey approach [79] is followed. A paradigmatic example of the development of a semantic theory of processes in which behavioural, axiomatic, and denotational semantics coexist harmoniously and may be used to highlight different aspects of process behaviors is the theory of testing equivalence developed by De Nicola and Hennessy [31, 42]. In this theory, a process can be characterized operationally in terms of its reaction to experiments, and denotationally as a so-called *acceptance tree* [41]. Acceptance trees allow one to fully describe the behaviour of a process while abstracting completely from the operational details of its interactions with all the possible testers. Moreover, the domain-theoretic properties of this model allow one to establish properties of the behavioural semantics that would be very difficult to derive using purely operational methods. (See, e.g., the results in [42, Section 4.5].)

To our mind, the coincidence of axiomatic, behavioural/operational, and denotational semantics enjoyed by the theory of processes presented in [42] not only reinforces the naturalness of the chosen notion of program semantics, but also allows one to make good use of the complementary benefits afforded by these semantic descriptions in establishing

* On leave from the School of Cognitive and Computing Sciences, University of Sussex, Brighton BN1 9QH, United Kingdom. Partially supported by HCM project EXPRESS.

† Partially supported by a grant from the Danish Research Council.

‡ Centre of the Danish National Research Foundation.

[1] In this paper, we shall use the term "axiomatic semantics" to denote the characterization of process semantics by means of (in)equationally based proof systems. This branch of process theory is referred to as algebraic semantics by some authors (see, e.g., the title of [64]). In this study, we prefer to reserve the term algebraic semantics for the approach to programming language semantics described in, e.g., [38, 42].

0890-5401/96 $18.00

properties of processes. However, developing these three views of processes for each process description language from scratch and proving their coincidence is hard, subtle work; in addition, to quote from [55], giving denotational semantics to programming languages using the Scott–Strachey approach "involves an armamentarium of mathematical weapons otherwise unfamiliar in Computer Sciences." We thus believe that it would be beneficial to develop systematic ways of giving denotational semantics to process description languages, following the Scott–Strachey approach, starting from their SOS description. Of course, this is only worthwhile if the denotational semantics produced by the proposed techniques is automatically guaranteed to be in agreement with the behavioural and axiomatic views of processes. In particular, we should like to generate a denotational semantics that matches exactly our operational intuition about process behaviour, i.e., that is *fully abstract*, in the sense of Milner and Plotkin [57, 58, 70, 87], with respect to a reasonable notion of behavioural semantics. This paper aims at making a small contribution in this direction.

1.1. *Results*

In this paper, we present a general way of giving denotational semantics to a class of languages equipped with an operational semantics that fits the GSOS format of Bloom, Istrail, and Meyer [20, 24]. The canonical model used for this purpose will be Abramsky's domain of synchronization trees $\mathscr{D}$ presented in [1], and the denotational semantics automatically generated by our methods will be guaranteed to be fully abstract with respect to the finitely observable part of the bisimulation preorder studied in, e.g., [1, 39, 46, 61, 94], Moreover, in the process of establishing the full abstraction result, we also present an algorithm, along the lines of those given in [5, 7], to generate a complete axiomatization of the bisimulation preorder, thus fulfilling our aim of giving behavioural, axiomatic, and denotational accounts of process behaviour that are in complete agreement. As a byproduct of our denotational semantics, we shall be able to establish very general results about the behavioural bisimulation preorder that would be hard to prove using purely operational definitions. (For an example, cf. Theorem 6.17.) This is one of the major theoretical advantages of having several complementary semantic views of a language; proofs of program properties that may be very involved or even hard to find in one semantics can be approached in a totally different way in another. On a more speculative note, the denotational semantics we propose may also have some practical interest. For example, powerful effective induction rules such as Scott Induction (see, e.g., [42, 53] for a discussion of this proof principle) become usable to reason about process (in)equalities, and proofs about processes can, at least in principle, be carried out within the kind of axiom systems supported by a tool such as LCF [35].

The class of GSOS systems we shall give denotational semantics to will have the structure of most standard process algebras (see, e.g., [12, 18, 47, 65]). They will consist of sets of operations to construct finite, acyclic process graphs and facilities for the recursive definition of behaviours. Borrowing a terminology introduced in [48] in the context of denotational semantics, we shall refer to these languages as *compact* GSOS languages. Their operational semantics will be given in terms of a variation on the standard model of labelled transition systems [51] that takes divergence information into account. This will be done in such a way that the bisimulation preorder is a precongruence with respect to all the operators in the language. In order to obtain this substitutivity result, special care must be taken in interpreting negative premises in GSOS rules; in particular, negative premises will only be interpreted over convergent (or fully specified) processes. (A similarly motivated choice is made in [89], where negative premises are only interpreted over stable processes, i.e., processes that cannot perform internal transitions.) Intuitively, this is because, in order to find out what a process can*not* do, we need to know precisely what its capabilities are, and the initial behaviour of a divergent process is only partially specified. A consequence of our choice is that, for example, the rule

$$\frac{x \overset{a}{\not\longrightarrow}}{f(x) \overset{a}{\longrightarrow} f(x)}$$

cannot be used to derive the result that the term $f(\Omega)$ has an a-labelled transition to itself, where Ω denotes the typical totally divergent process with no transitions. As a byproduct of our approach, we are able to give a simple semantics to GSOS languages in which negative premises are allowed to coexist with unguarded recursive definitions. This contrasts with the standard GSOS semantics given in [20, 24] in which the interplay between negative premises and unguarded recursion may lead to the operational specification of a language without a well-defined operational semantics. (See, e.g., [20, 36] and Section 4 of this paper for an example.)

Our first main result is that, with our choice of operational semantics for GSOS languages, the bisimulation preorder is substitutive with respect to all language contexts. (See Theorem 3.9 and Theorem 4.8.) Moreover, as a consequence of general results established by Abramsky in [1], we are able to give a characterization of the finitely observable (or finitary) part of the bisimulation preorder for every GSOS language. Intuitively, this is the preorder obtained by restricting the prebisimulation relation to observations of finite depth.

We then show how to automatically give a denotational semantics for a GSOS language in terms of Abramsky's

domain of synchronization trees $\mathscr{D}$. To this end, following the ideas of initial algebra semantics [30, 34, 59, 78], it is sufficient to endow Abramsky's model with an appropriate continuous algebra structure in the sense of [34, 38, 42]. This we do by showing how the GSOS rules defining the operational semantics of an operation symbol f of a compact GSOS language can be used to define a continuous function $\mathbf{f}_{\mathscr{D}}$ of the appropriate arity over the domain of synchronization trees $\mathscr{D}$. In defining the semantic counterparts of the operations in a compact GSOS language, we shall rely on a description of the domain $\mathscr{D}$ presented in [48], where it is shown how to reconstruct $\mathscr{D}$ from a suitable preorder over finite synchronization trees. This view of $\mathscr{D}$ will allow us to define each semantic operation $\mathbf{f}_{\mathscr{D}}$ in stepwise fashion from monotonic operations over finite synchronization trees. We hope that this choice will make the presentation more accessible to readers who are unfamiliar with domain theory [71, 86].

As a result of our general framework, we shall then show that the denotational semantics so obtained is guaranteed to be in complete agreement with the chosen behavioural semantics. More precisely, for every compact GSOS language, the denotational semantics produced by the general approach presented in this paper is always fully abstract with respect to the finitary part of the bisimulation preorder. The key to the proof of this result is a general theorem that states that, for every compact GSOS language, the finitary part of the bisimulation preorder is completely determined by how it acts on recursion-free processes. Relations that have this property are called *algebraic* in [42]. The proof of the algebraicity of the behavioural preorder is rather involved, and relies on an algorithm for generating an inequational theory that is partially complete with respect to the bisimulation preorder, in the sense of [39], for arbitrary compact GSOS systems. The partially complete axiomatization generated by our methods proves exactly all the valid inequalities of the form $P \leqslant Q$, where P is a recursion-free term and Q is any term. It can be lifted to the whole of a compact GSOS language by adding to the proof system a very powerful induction rule, called ω-induction in [42].

1.2. *Related Work*

The work reported in this paper is by no means the first attempt to systematically derive denotational models from SOS language specifications. The main precursors to this study in the field of the meta-theory of process description languages may be found in the work of Bloom [21] and of Rutten and Turi [74–77]. In an unpublished paper [21], Bloom gives operational, logical, relational, and three denotational semantics for GSOS languages without negative premises and unguarded recursion, and shows that they coincide. Bloom's work is based on the behavioural notion of simulation [44], and two of his denotational semantics are given in terms of Scott domains based on finite synchronization trees. On the other hand, the work by Rutten presented in [74–76] gives methods for deriving denotational semantics based on complete metric spaces and Aczel's non-well-founded sets [10] for languages specified in terms of sub-formats of the *tyft/tyxt* format due to Groote and Vaandrager [37]. In particular, the reference [76] gives a detailed and clear introduction to a technique, called *processes as terms* by Rutten, for the definition of operations on semantic models from operational rules. Rutten's general "processes as terms" approach could have been applied to yield an equivalent formulation of the semantic operations on finite synchronization trees we present in Section 6.2, provided we allowed for GSOS languages with a denumerable set of constant symbols. In this study, however, we have plumped for the more direct construction of the operations given in Definition 6.9. The work presented in the aforementioned papers by Rutten has recently been generalized by Rutten and Turi in [77]. In that paper, the authors show how to give denotational semantics to languages specified by transition system specifications in full *tyft/tyxt* format [37], and investigate in a categorical perspective the essential properties of semantic domains that make their definitions possible.

In [3], various notions of process observations are considered in a uniform algebraic framework provided by the theory of *quantales* (see, e.g., [73]). The methods developed by Abramsky and Vickers in [3] yield, in a uniform fashion, observational logics and denotational models for each notion of process observation they consider. Their work is, however, semantic in nature and ignores the algebraic structure of process expressions.

In the area of the semantics of functional programs, developments that are somewhat similar in spirit to those pursued in this study are presented by Smith in [82]. In that paper, Smith studies a natural notion of preorder over programs written in a simple functional programming language, and shows how any ordering on programs with certain basic properties can be extended to a term model that is fully abstract with respect to it.

Finally, it is hard to underestimate the debt that our work owes to the pioneering work of Abramsky, Hennessy, Milner, Plotkin and their coworkers in the field of denotational models for concurrency. Without the inspiration of seminal papers such as [1, 41, 43, 45, 56], this work would simply not have been possible.

1.3. *Outline of the Paper*

The paper is organized as follows. Section 2 presents the basic notions on transition systems and prebisimulation that will be needed in this study. We then go on to present GSOS languages and rules; first we discuss recursion-free

languages in Section 3, and present our new transition systems semantics for them. The main result of Section 3 is that our operational view of GSOS languages induces operations that are substitutive with respect to the bisimulation preorder. Section 4 introduces GSOS languages with recursion and their operational semantics. There we show how to apply our approach to give reasonable operational semantics to languages combining operations defined using negative premises with arbitrary recursive definitions. Section 5 is devoted to background material on algebraic and denotational semantics needed for the remainder of the paper. Compact GSOS languages are introduced in Section 6.1. Our method for giving a denotational semantics to arbitrary compact GSOS languages is presented in Section 6.2, where the proof of full abstraction of the resulting semantics is also given. Finally, we present in Section 7 an algorithm that, for any compact GSOS language, generates an inequational theory that is partially complete with respect to the bisimulation preorder. The paper concludes with some remarks on our work and a mention of topics for further research.

As this is not an introductory paper on the meta-theory of process description languages, we have taken the liberty of referring the readers to other publications in the literature for motivations and some of the background technical material. We hope, however, that our choice of presentation will make the paper accessible to uninitiated readers.

To improve the readability of the paper and keep its size manageable, we have also chosen to omit some of the proofs of results of purely technical interest. These may all be found in the report version of this paper [9] that is electronically available at the address given in the bibliography.

2. PRELIMINARIES ON LABELLED TRANSITION SYSTEMS

We begin by reviewing the basic notions on transition systems that will be needed in this study. The interested reader is invited to consult, e.g., [51, 72] for more details and extensive motivations.

The operational semantics of the languages considered in this paper will be given in terms of a variation on the model of labelled transition systems [51] that takes divergence information into account. We refer the interested readers to, e.g., [39, 46, 61, 94] for motivation and more information on (variations on) this semantic model for reactive systems.

DEFINITION 2.1 (Labelled Transition Systems with Divergence). A *labelled transition system with divergence* (*lts*) is a quadruple $(\mathsf{P}, \mathsf{Lab}, \rightarrow, \uparrow)$, where:

- P is a set of *processes*, ranged over by s, t;
- Lab is a set of *labels*, ranged over by l;
- $\rightarrow \subseteq \mathsf{P} \times \mathsf{Lab} \times \mathsf{P}$ is a *transition relation* (as usual, we shall use the more suggestive notation $s \xrightarrow{l} t$ in lieu of $(s, l, t) \in \rightarrow$);
- $\uparrow \subseteq \mathsf{P}$ is a *divergence predicate*, notation $s\uparrow$.

We write $s\downarrow$, read "s definitely converges," iff it is not the case that $s\uparrow$, and $s \rightarrow t$ iff $s \xrightarrow{a} t$ for some $a \in \mathsf{Lab}$. We shall use the symbol $\rightarrow^*$ to stand for the reflexive and transitive closure of $\rightarrow$. The *sort* of a process s is defined as follows:

$$\mathsf{sort}(s) = \{a \in \mathsf{Lab} \mid \exists t, u : s \rightarrow^* t \xrightarrow{a} u\}.$$

An lts is *sort-finite* iff $\mathsf{sort}(s)$ is finite for every process s.

A useful source of examples for labelled transition systems with divergence is the set of finite synchronization trees over a set of labels Lab, denoted by $\mathsf{ST(Lab)}$. These are the sets generated by the inductive definition

$$\frac{\{l_i \in \mathsf{Lab}, t_i \in \mathsf{ST(Lab)}\}_{i \in I}}{\{\langle l_i, t_i \rangle \mid i \in I\}[\cup\{\bot\}] \in \mathsf{ST(Lab)}}$$

where I is a finite index set, and the notation $[\cup\{\bot\}]$ means optional inclusion of $\bot$. As will become clear in a moment, the symbol $\bot$ will be used to represent the fact that a synchronization tree is divergent. The set of finite synchronization trees $\mathsf{ST(Lab)}$ can be turned into a labelled transition system with divergence by stipulating that, for $t \in \mathsf{ST(Lab)}$:

- $t\uparrow$ iff $\bot$ is in t, and
- $t \xrightarrow{l_i} t_i$ iff $\langle l_i, t_i \rangle$ is in t.

The behavioural relation over processes that we shall study in this paper is that of *prebisimulation* [39, 46, 61, 94] (also known as *partial bisimulation* [1]).

DEFINITION 2.2 (Prebisimulation). Let $\Lambda = (\mathsf{P}, \mathsf{Lab}, \rightarrow, \uparrow)$ be an lts. Let $\mathsf{Rel(P)}$ denote the set of binary relations over P. Define the functional $\mathscr{F} : \mathsf{Rel(P)} \rightarrow \mathsf{Rel(P)}$ by:

$$\mathscr{F}(\mathfrak{R}) = \{(s, t) \mid \forall l \in \mathsf{Lab}$$
$$\bullet\ s \xrightarrow{l} s' \Rightarrow \exists t' : t \xrightarrow{l} t' \text{ and } s' \mathfrak{R} t'$$
$$\bullet\ s\downarrow \Rightarrow t\downarrow \text{ and } [t \xrightarrow{l} t' \Rightarrow \exists s' : s \xrightarrow{l} s' \text{ and } s' \mathfrak{R} t']\}.$$

A relation $\mathfrak{R}$ is a prebisimulation iff $\mathfrak{R} \subseteq \mathscr{F}(\mathfrak{R})$. We write $s_1 \lesssim_\Lambda s_2$ iff there exists a prebisimulation $\mathfrak{R}$ such that $s_1 \mathfrak{R} s_2$. (The subscript Λ will be omitted when this causes no confusion.)

The relation $\lesssim$ is a preorder over P based on a variation on bisimulation equivalence [65, 68]. Its kernel will be denoted by $\sim$, i.e., $\sim = \lesssim \cap \lesssim^{-1}$. Intuitively, $s_1 \lesssim s_2$ if s_2's behaviour is at least as specified as that of s_1, and s_1 and s_2

can simulate each other when restricted to the part of their behaviour that is fully specified. A divergent state s with no outgoing transition is a minimal element with respect to $\lesssim$, and intuitively corresponds to a process whose behaviour is totally unspecified—essentially an operational version of the bottom element $\perp$ in Scott's theory of domains [71, 79, 86].

Although the relations $\lesssim$ and $\sim$ have been defined over a given lts, we often want to use them to compare processes from different lts's; for example, we shall often compare states in an lts with finite synchronization trees. This can be done in standard fashion by forming the disjoint union of the two systems, and then using $\lesssim$ and $\sim$ on the resulting lts. In the sequel, this will be done without further comment.

Notation 2.3. The largest prebisimulation over an lts obtained as the disjoint union of an lts A and the lts of finite synchronization trees will be denoted by $\lesssim_A$ throughout the paper.

In this study, we shall be interested in relating the notion of prebisimulation to a preorder on processes induced by a denotational semantics given in terms of an algebraic domain [71, 86]. As such preorders are completely determined by how they act on *finite processes*, we shall be interested in comparing them with the "finitely observable", or *finitary*, part of the bisimulation in the sense of, e.g., [38, 39]. The following definition is from [1].

DEFINITION 2.4. The *finitary preorder* $\lesssim^{\mathrm{F}}$ is defined on any lts by

$$s \lesssim^{\mathrm{F}} s' \Leftrightarrow \forall t \in \mathsf{ST(Lab)}.\, t \lesssim s \Rightarrow t \lesssim s'.$$

An alternative method for using the functional $\mathscr{F}$ to obtain a behavioural preorder is to apply it inductively as follows:

- $\lesssim_0 = \mathsf{P} \times \mathsf{P}$,
- $\lesssim_{n+1} = \mathscr{F}(\lesssim_n)$

and finally $\lesssim_\omega = \bigcap_{n \geqslant 0} \lesssim_n$. Intuitively, the preorder $\lesssim_\omega$ is obtained by restricting the prebisimulation relation to observations of finite depth. The preorders $\lesssim$, $\lesssim_\omega$, and $\lesssim^{\mathrm{F}}$ are, in general, related thus:

$$\lesssim \subseteq \lesssim_\omega \subseteq \lesssim^{\mathrm{F}}.$$

Moreover the inclusions are, in general, strict. The interested reader is referred to [1] for a wealth of examples distinguishing these preorders, and a very deep analysis of their general relationships and properties. Here we simply state the following useful result, which is a simple consequence of [1, Lemma 5.10]:

LEMMA 2.5. *Let* $(\mathsf{P}, \mathsf{Lab}, \rightarrow, \uparrow)$ *be an lts with divergence. Then, for every* $t \in \mathsf{ST(Lab)}$, $s \in \mathsf{P}$, $t \lesssim s$ *iff* $t \lesssim_\omega s$.

3. GSOS LANGUAGES

We assume some familiarity with process algebra and structural operational semantics (see, e.g., [18, 20, 24, 37, 42, 47, 65, 72] for more details and extensive motivations).

Let Var be a denumerable set of *meta-variables* ranged over by x, y. A *signature* Σ consists of a set of *operation symbols*, disjoint from Var, together with a function *arity* that assigns a natural number to each operation symbol. Throughout this paper, following the standard lines of algebraic semantics (see, e.g., [38, 42]), we shall assume that a signature contains a distinguished function symbol Ω of arity zero to denote the totally unspecified, or divergent, process, i.e., a process about whose behaviour we have no information.[2] The set $\mathbb{T}(\Sigma, \mathsf{Var})$ of *terms* over Σ and Var (abbreviated to $\mathbb{T}(\Sigma)$ when the set of variables is clear from the context or immaterial) is the least set such that

- Each $x \in \mathsf{Var}$ is a term.
- If f is an operation symbol of arity l, and $P_1, ..., P_l$ are terms, then $f(P_1, ..., P_l)$ is a term.

We shall use P, Q, ... to range over terms and the symbol $\equiv$ for the relation of syntactic equality on terms. $\mathrm{T}(\Sigma)$ is the set *closed* terms over Σ, i.e., terms that do not contain variables. Constants, i.e., terms of the form $f(\,)$, will be abbreviated as f.

A *Σ-context* $C[\mathbf{x}]$ is a term in which at most the variables $\mathbf{x}$ appear. $C[\mathbf{P}]$ is $C[\mathbf{x}]$ with x_i replaced by P_i wherever it occurs. We say that a relation $\mathfrak{R} \subseteq \mathrm{T}(\Sigma) \times \mathrm{T}(\Sigma)$ is *closed with respect to Σ-contexts* iff for every Σ-context $C[\mathbf{x}]$ and vectors of closed terms $\mathbf{P}$ and $\mathbf{Q}$ of the appropriate length

$$\mathbf{P}\ \mathfrak{R}\ \mathbf{Q} \text{ implies } C[\mathbf{P}]\ \mathfrak{R} C[\mathbf{Q}],$$

where the relation $\mathfrak{R}$ is extended pointwise to vectors of equal length.

Besides terms we have *actions*, elements of some given finite set Act, which is ranged over by a, b, c. A *positive transition formula* is a triple of two terms and an action, written $P \xrightarrow{a} P'$. A *negative transition formula* is a pair of a term and an action, written $P \overset{a}{\not\longrightarrow}$. In general, the terms in the transition formula will contain variables. Transition formulae will be ranged over by φ.

DEFINITION 3.1 (GSOS Rules). Suppose Σ is a signature. A *GSOS rule* r over Σ is an inference rule of the form

$$\frac{\left[\begin{array}{l} \bigcup_{i=1}^{l} \{x_i \xrightarrow{a_{ij}} y_{ij} \mid 1 \leqslant j \leqslant m_i\} \\ \quad \cup \bigcup_{i=1}^{l} \{x_i \overset{b_{ik}}{\not\longrightarrow} \mid 1 \leqslant k \leqslant n_i\} \end{array}\right]}{f(x_1, ..., x_l) \xrightarrow{c} C[\mathbf{x}, \mathbf{y}]} \qquad (1)$$

[2] In fact, as will become clear in the remainder of this paper, Ω is just syntactic sugar for the recursive term $\mathrm{fix}(X = X)$. See Section 4 for details.

where all the variables are distinct, $m_i, n_i \geqslant 0$, f is an operation symbol from Σ with arity l, $C[\mathbf{x}, \mathbf{y}]$ is a Σ-context, and the a_{ij}, b_{ik} and c are actions in Act.

It is useful to name components of rules of the form (1). The operation symbol f is the *principal operation* of the rule, and the term $f(\mathbf{x})$ is the *source*. $C[\mathbf{x}, \mathbf{y}]$ is the *target* (sometimes denoted by target(r)); c is the *action* (sometimes denoted by action(r)); the formulae above the line are the *antecedents*; and the formula below the line is the *consequent*. If, for some i, $m_i > 0$ then we say that r *tests its ith argument positively*. Similarly if $n_i > 0$ then we say that r *tests its ith argument negatively*. An operation f *tests its ith argument positively* (resp. *negatively*) if it occurs as principal operation of a rule that tests its ith argument positively (resp. negatively). We say that an operation f *tests its ith argument* if it tests it either positively or negatively.

DEFINITION 3.2 (GSOS Systems). A *GSOS system* is a pair $G = (\Sigma_G, R_G)$, where Σ_G is a finite signature and R_G is a finite set of GSOS rules over Σ_G containing no rules with Ω as principal operation.

EXAMPLE. An example of GSOS system, the language preACP$_{\Omega\theta}$,[3] is presented in Fig. 1. We shall use this concrete language as a running example throughout the paper to illustrate our definitions and results.

The language preACP$_{\Omega\theta}$ is a variation on ACP [18] with action prefixing in lieu of general sequential composition. Its parallel composition operator, denoted by $\|$, is parameterized with respect to a partial, commutative, and associative communication function γ: Act $\times$ Act $\rightharpoonup$ Act. An operation in preACP$_{\Omega\theta}$ that uses the power of negative premises, at least in the presence of a non-trivial priority structure on actions, is the priority operation θ of Baeten *et al.* [16]. In order to define this operation, we assume a given partial ordering relation $>$ on Act. Intuitively $b > a$ is interpreted as "action b has priority over action a."

The sub-language of preACP$_{\Omega\theta}$ consisting only of the operations Ω, δ, $a._-$, and $+$ will be denoted by FINTREE$_\Omega$. We shall use the standard process algebra conventions for the FINTREE$_\Omega$ language. For example, for $I = \{i_1, ..., i_n\}$ a finite index set, we write $\sum_{i \in I}$ for $P_{i_1} + \cdots + P_{i_n}$. By convention $\sum_{i \in \varnothing} P_i$ stands for δ. (*End of Example*)

GSOS systems have been introduced and studied in depth in [20, 24]. Intuitively, a GSOS system gives a language, whose constructs are the operations in the signature Σ_G, together with a Plotkin-style structural operational semantics [72] for it defined by the set of conditional rules R_G. In this study, the operational semantics of a GSOS system will be given in terms of labelled transition systems with

[3] Here we follow the spirit of the terminology suggested in [15].

Signature	*Arity*	*Rules*
δ	0	no rules
Ω	0	no rules
$a._-$ $(a \in \mathsf{Act})$	1	$a.x \xrightarrow{a} x$
$+$	2	$\frac{x \xrightarrow{a} x'}{x+y \xrightarrow{a} x'}$ $\quad$ $\frac{y \xrightarrow{a} y'}{x+y \xrightarrow{a} y'}$
$\|$	2	$\frac{x \xrightarrow{a} x'}{x\|y \xrightarrow{a} x'\|y}$ $\quad$ $\frac{y \xrightarrow{a} y'}{x\|y \xrightarrow{a} x\|y'}$ $\quad$ $\frac{x \xrightarrow{a} x', y \xrightarrow{b} y'}{x\|y \xrightarrow{c} x'\|y'}$ $\gamma(a,b) \simeq c$
θ	1	$\frac{x \xrightarrow{a} x', (\forall b > a) x \xrightarrow{b} \not\to}{\theta(x) \xrightarrow{a} \theta(x')}$

FIG. 1 The language preACP$_{\Omega\theta}$

divergence. In order to obtain this non-standard interpretation, we aim at using the rules in a GSOS system G to define a divergence predicate over terms and a transition relation in such a way that our definitions:

1. specialize to those originally given by Bloom, Istrail, and Meyer in their seminal studies [20, 24] when divergence is not taken into account;
2. give results that are in agreement with those already presented in the literature when applied to standard process description languages; and
3. produce operators that are well-behaved with respect to the notion of prebisimulation, i.e., operations for which prebisimulation is a precongruence.

First of all, we shall use the rules in a GSOS systems to define a divergence (or under-specification) predicate on the set of closed terms over Σ_G. In fact, as is common practice in the literature on process algebras, we shall define the notion of convergence, and use it to define the divergence predicate we are after. Intuitively, a term P is convergent if the set of its initial transitions is fully specified. The basic divergent term is Ω, the totally unspecified process. A term of the form $f(\mathbf{P})$ is convergent iff the set of its initial transitions only depends on those arguments P_is whose initial behaviour is completely known. This informal discussion motivates the following definition.

DEFINITION 3.3 (Convergence). Let $G = (\Sigma_G, R_G)$ be a GSOS system. The convergence predicate $\downarrow_G$ (abbreviated to $\downarrow$ when the GSOS system G is clear from the context) is the least predicate over $T(\Sigma_G)$ that satisfies the following clause: $f(P_1, ..., P_l) \downarrow_G$ if

1. $f \neq \Omega$, and
2. for every argument i of f, if f tests i then $P_i \downarrow_G$.

We write $P \uparrow_G$ iff it is not the case that $P \downarrow_G$.

When applied to the language preACP$_{\Omega\theta}$, Definition 3.3 gives the following convergence predicate:

- $\delta\downarrow$,
- if $P\downarrow$ and $Q\downarrow$, then $P+Q\downarrow$ and $P \parallel Q\downarrow$,
- if $P\downarrow$ then $\theta(P)\downarrow$.

The reader familiar with the literature on prebisimulation over CCS-like languages will have noted that the above definition generalizes those given in, e.g., [31, 34, 46]. For instance, when applied to the recursion-free fragment of the version of Milner's SCCS considered in [39], it delivers exactly Hennessy's convergence predicate.

We shall now present our non-standard operational semantics for GSOS languages. As stated above, we take as our starting point the original theory developed by Bloom, Istrail, and Meyer. Informally, the original intent of a GSOS rule is as follows. Suppose that we are wondering whether $f(\mathbf{P})$ is capable of taking a c-step. We look at each rule with principal operation f and action c in turn. We inspect each positive antecedent $x_i \xrightarrow{a_{ij}} y_{ij}$, checking if P_i is capable of taking an a_{ij}-step for each j and if so calling the a_{ij}-children Q_{ij}. We also check the negative antecedents; if P_i is *in*capable of taking a b_{ik}-step for each k. If so, then the rule *fires* and $f(\mathbf{P}) \xrightarrow{c} C[\mathbf{P}, \mathbf{Q}]$. Roughly, this means that the transition relation associated with a GSOS system in [24] is the one defined by structural induction on terms using the rules in R_G.

In the presence of divergence information, we shall define the transition relation over terms in a similar vein. However, we shall interpret negative transition formulae *over convergent processes only*. Intuitively, to know that a process cannot initially perform a given action, we need to find out precisely all the actions that it can perform. If a process is divergent, its set of initial actions is not fully specified; thus we cannot be sure whether such a process satisfies a negative transition formula or not.

For the sake of completeness, we shall now formally define the lts with divergence induced by a GSOS system following [7, 20, 24, 47]. (Our presentation most closely follows the one given in [7].)

DEFINITION 3.4. A (*closed*) *Σ-substitution* is a function σ from variables to (closed) terms over the signature Σ. For each term P, $P\sigma$ will denote the result of substituting $\sigma(x)$ for each x occurring in P. For t a term, transition formula, GSOS rule, etc., we write $t\sigma$ for the result of substituting $\sigma(x)$ for each x occurring in t.

The notation $\{P_1/x_1, ..., P_n/x_n\}$, where the P_is are terms and the x_i's are distinct variables, will often be used to denote the substitution that maps each x_i to P_i, and leaves all the other variables unchanged.

DEFINITION 3.5. Let G be a GSOS system with derived convergence predicate $\downarrow_G$. A transition relation over the signature Σ_G is a relation $\rightsquigarrow \subseteq T(\Sigma_G) \times \mathsf{Act} \times T(\Sigma_G)$.

Let $\rightsquigarrow$ be a transition relation and σ a closed substitution. For each transition formula φ, the predicate $\rightsquigarrow, \sigma \models \varphi$ is defined by

$$\rightsquigarrow, \sigma \models P \xrightarrow{a} A \triangleq P\sigma \overset{a}{\rightsquigarrow} Q\sigma$$

$$\rightsquigarrow, \sigma \models P \not\xrightarrow{a} \triangleq P_\sigma\downarrow_G \qquad \text{and} \qquad \nexists Q \in T(\Sigma_G)\colon P\sigma \overset{a}{\rightsquigarrow} Q$$

For H a set of transition formulae, we define

$$\rightsquigarrow, \sigma \models H \triangleq \forall \varphi \in H\colon \rightsquigarrow, \sigma \models \varphi$$

and for $r = H/\varphi$ a GSOS rule of the form (1),

$$\rightsquigarrow, \sigma \models \frac{H}{\varphi} \triangleq (\rightsquigarrow, \sigma \models H \Rightarrow \rightsquigarrow, \sigma \models \varphi).$$

To exemplify the definition of the notion of satisfaction of transition formulae given above, let us consider the language preACP$_{\Omega\theta}$ and the substitution $\sigma = \{a.\delta + \Omega/x, \delta/y\}$. Then, for every transition relation $\rightsquigarrow$ and action $b \in \mathsf{Act}$,

$$\rightsquigarrow, \sigma \models x \xrightarrow{b} y \Leftrightarrow a.\delta + \Omega \overset{b}{\rightsquigarrow} \delta.$$

On the other hand, regardless of the properties of the transition relation $\rightsquigarrow$, $\rightsquigarrow, \sigma \not\models x \not\xrightarrow{b}$ because $a.\delta + \Omega$ is a divergent term.

The reader familiar with the literature on Hennessy–Milner logics [44] for prebisimulation-like relations will have noted that our notion of satisfaction for negative transition formulae is akin to that for formulae of the form $[a]F$ given in, e.g., [1, 8, 61, 84, 85]. In those references, the new interpretation is necessary to obtain monotonicity of the satisfaction relation with respect to the appropriate notion of prebisimulation. In this study, our interpretation of negative premises will be crucial to obtain operations that are monotonic with respect to the notion of prebisimulation. (See, e.g., Theorem 3.9). Basically, it will ensure that, for a closed term P, the transition formula $P \xrightarrow{a}$ holds iff $Q \not\xrightarrow{a}$ holds for every closed term with $P \lesssim Q$.

DEFINITION 3.6. Suppose G is a GSOS system and $\rightsquigarrow$ is a transition relation over Σ_G. Then $\rightsquigarrow$ is sound for G iff for every rule $r \in R_G$ and every closed Σ_G-substitution σ, we have $\rightsquigarrow, \sigma \models r$. A transition $P \overset{a}{\rightsquigarrow} Q$ is *supported* by some rule $H/\varphi \in R_G$ iff there exists a substitution σ such that $\rightsquigarrow, \sigma \models H$ and $\varphi\sigma = (P \xrightarrow{a} Q)$. The relation $\rightsquigarrow$ is supported by G iff each transition in $\rightsquigarrow$ is supported by a rule in R_G.

The requirements of soundness and supportedness are sufficient to associate a unique transition relation with each GSOS system.

LEMMA 3.7. *For each GSOS system G there is a unique sound and supported transition relation.*

Proof. The unique sound and supported transition relation associated with a GSOS system G is the one defined by structural recursion on terms using the rules in R_G, with the proviso that, as formalized in Definition 3.5, negative transition formulae can only be satisfied by convergent terms. The interested reader is referred to the proof of Proposition 4.3 for details. ▮

We write $\rightarrow_G$ for the unique sound and supported transition relation for G.

LEMMA 3.8. *Suppose G is a GSOS system. Then the transition relation $\rightarrow_G$ is finitely branching, i.e., for every $P \in T(\Sigma_G)$, the set* $\mathsf{Der}(P) = \{(a, Q) \mid P \xrightarrow{a}_G Q\}$ *is finite.*

Proof. A minor modification of the proof of the standard result for GSOS languages in [20]. ▮

The lts with divergence specified by a GSOS system G is then given by

$$\mathsf{lts}(G) = (\mathrm{T}(\Sigma_G), \rightarrow_G, \uparrow_G).$$

The largest prebisimulation over $\mathsf{lts}(G)$ will be denoted by $\lesssim_G$, and its kernel by $\sim_G$. (The subscript G will be omitted from these relations when this causes no confusion).

EXAMPLE. We exemplify our approach using our running example, the language in Fig. 1, by considering some identities involving simple terms that use the priority operation θ.

The term $\theta(\Omega)$ is divergent, as Ω is. Moreover, it has no transition because Ω has none. We thus have that $\theta(\Omega) \sim \Omega$.

Consider a term of the form $P \equiv a.\delta + \Omega$, with a a maximal element in the poset $(\mathsf{Act}, >)$, i.e., with a an action with maximal priority. Then the rule for θ with action a has no negative antecedents, and it can be used to establish the transition $\theta(P) \xrightarrow{a} \theta(\delta)$. Indeed, this is the only transition that is possible from $\theta(P)$. Because $\theta(P)$ is divergent, as P is, it is easy to see that $\theta(P) \sim a.\delta + \Omega$.

On the other hand, if a is *not* maximal in the poset $(\mathsf{Act}, >)$, the rule for θ with action a will have at least one negative antecedent. As P is divergent, that rule cannot be used to derive a transition from the term $\theta(P)$. It thus follows that, for such a term P, $\theta(P) \sim \Omega$. (*End of Example*)

We are now ready to establish the first main result of this paper. Namely, we shall prove that the operations of a GSOS system preserve the semantic notion of prebisimulation. This is the import of the following theorem.

THEOREM 3.9. *Let G be a GSOS system. Then $\lesssim_G$ is a precongruence for all operation symbols f of G, i.e., $\mathbf{P} \lesssim_G \mathbf{Q} \Rightarrow f(\mathbf{P}) \lesssim_G f(\mathbf{Q})$.*

Proof. Consider the least relation $\mathfrak{R}$ satisfying the following conditions:

1. $\lesssim_G \subseteq \mathfrak{R}$, and
2. if $P_i \mathfrak{R} Q_i$ for $1 \leqslant i \leqslant l$, then $f(\mathbf{P}) \mathfrak{R} f(\mathbf{Q})$.

Note that $\mathfrak{R}$ so defined is the smallest relation that contains $\lesssim_G$ and is closed with respect to all Σ_G-contexts.

The statement of the theorem follows immediately if we prove that $\mathfrak{R}$ is a prebisimulation. This we show by induction on the definition of the relation $\mathfrak{R}$. Assume thus that $P \mathfrak{R} Q$. By the definition of $\mathfrak{R}$, we have that either

- $P \lesssim_G Q$, or
- $P \equiv f(\mathbf{P})$, $Q \equiv f(\mathbf{Q})$ and $\mathbf{P} \mathfrak{R} \mathbf{Q}$, for some $\mathbf{P}$, $\mathbf{Q}$.

If $P \lesssim_G Q$, then the clauses of Definition 2.1 are trivially met as $\lesssim_G$ is itself a prebisimulation included in $\mathfrak{R}$. We shall thus concentrate on showing that they are met when $P \equiv f(\mathbf{P})$, $Q \equiv f(\mathbf{Q})$, and $\mathbf{P} \mathfrak{R} \mathbf{Q}$, for some $\mathbf{P}$, Q, under the inductive hypothesis that they are met for each pair of terms $P_i \mathfrak{R} Q_i$.

We check that each clause of the definition of prebisimulation is met in turn.

1. Assume that $P \equiv f(\mathbf{P}) \xrightarrow{c}_G R$ for some $R \in \mathrm{T}(\Sigma_G)$. We shall show that $Q \equiv f(\mathbf{Q}) \xrightarrow{c}_G S$ for some S such that $R \mathfrak{R} S$.

As $f(\mathbf{P}) \xrightarrow{c}_G R$ and $\rightarrow_G$ is supported by R_G, there exist a rule r for f of the form (1) and a closed substitution σ such that:

(a) $f(\mathbf{x})\,\sigma = f(\mathbf{P})$, i.e., $\sigma(x_i) = P_i$ for every $i \in \{1, ..., l\}$;

(b) $C[\mathbf{x}, \mathbf{y}]\,\sigma = R$;

(c) for every $1 \leqslant i \leqslant l$, $1 \leqslant j \leqslant m_i$, $\sigma(x_i) = P_i \xrightarrow{a_{ij}}_G \sigma(y_{ij})$; and

(d) for every $1 \leqslant i \leqslant l$ with $n_i > 0$, $\sigma(x_i) = P_i \downarrow_G$ and, for every $1 \leqslant k \leqslant n_i$, $P_i \not\xrightarrow{b_{ik}}_G$.

We aim at using r to construct a matching transition from $f(\mathbf{Q})$ with respect to $\mathfrak{R}$. This requires checking that all the antecedents of rule r are suitably met by $\mathbf{Q}$.

We examine the positive antecedents first. As $P_i \mathfrak{R} Q_i$ for each $1 \leqslant i \leqslant l$, by induction we have that for every $1 \leqslant i \leqslant l$, $1 \leqslant j \leqslant m_i$, there exists a term S_{ij} such that $Q_i \xrightarrow{a_{ij}}_G S_{ij}$, and $\sigma(y_{ij}) \mathfrak{R} S_{ij}$. This implies that the positive antecedents of r can be met by using the substitution $\tau = \{\mathbf{Q}/\mathbf{x}, \mathbf{S}/\mathbf{y}\}$.

As $P_i \mathfrak{R} Q_i$ and $P_i \downarrow_G$ if $n_i > 0$ $(1 \leqslant i \leqslant l)$, another application of the inductive hypothesis gives that for every $1 \leqslant i \leqslant l$ with $n_i > 0$, $\tau(x_i) = Q_i \downarrow_G$ and $\tau(x_i) = Q_i \not\xrightarrow{b_{ik}}_G$, $1 \leqslant k \leqslant n_i$, i.e., all the negative antecedents of rule r can also be met by the substitution τ.

Thus the substitution τ and the rule r can be used to derive the transition

$$f(\mathbf{Q}) \xrightarrow{c}_G C[\mathbf{x}, \mathbf{y}]\,\tau.$$

We are now left to show that $R = C[\mathbf{x}, \mathbf{y}]\ \sigma\ \mathfrak{R}\ C[\mathbf{x}, \mathbf{y}]\ \tau$. However, as by construction $\mathfrak{R}$ is closed with respect to all Σ_G-contexts, this is immediate from the fact that, for every variable z occurring in $\mathbf{x}$ or $\mathbf{y}$, $\sigma(z)\ \mathfrak{R}\tau(z)$.

2. We now show that $f(\mathbf{P})\downarrow_G$ implies $f(\mathbf{Q})\downarrow_G$. Assume that $f(\mathbf{P})\downarrow_G$. By the definition of the predicate $\downarrow_G$, this is because $f \neq \Omega$ and, for every argument i of f, f tests i implies that $P_i\downarrow_G$. As $P_i\ \mathfrak{R}\ Q_i$, the induction hypothesis gives that $Q_i\downarrow_G$ for every i tested by f. Thus $f(\mathbf{Q})\downarrow_G$.

3. Assume that $f(\mathbf{P})\downarrow_G$, $f(\mathbf{Q})\downarrow_G$ and $f(\mathbf{Q}) \xrightarrow{c}_G S$. In this case, following the lines of point 1 above, it is not hard to show that $f(\mathbf{P}) \xrightarrow{c}_G R$ for some term R such that $R\ \mathfrak{R}\ S$.

This proves that R is a prebisimulation. ▮

It is interesting to remark here that the above theorem would *not* hold if we allowed negative premises to be satisfied by divergent terms. As a simple example of this fact, consider the operation θ in our running example, and let us assume that the poset $(\mathsf{Act}, >)$ is non-trivial. Let a be an action that is *not* maximal in the poset $(\mathsf{Act}, >)$, and consider the term $P \equiv a.\delta + \Omega$. If we were allowed to interpret negative premises over divergent terms, then the rule for θ with action a could be used to derive the transition $\theta(P) \xrightarrow{a} \theta(\delta)$. However, any term of the form $a.\delta + b.\delta$ with $b > a$ (these terms exist as a is not maximal in $(\mathsf{Act}, >)$) would have the properties that:

1. $P \lesssim a.\delta + b.\delta$, and
2. $\theta(a.\delta + b.\delta) \not\xrightarrow{a}$.

This would imply that $\theta(P) \not\lesssim \theta(a.\delta + b.\delta)$.

4. GSOS SYSTEMS WITH RECURSION

In this section we consider GSOS languages that include explicit recursive definitions of processes. Let $G = (\Sigma_G, R_G)$ be a GSOS system, and let PVar be a fresh denumerable set of *process variables* ($X, Y \in \mathsf{PVar}$). The set of recursive terms over Σ_G and PVar, denoted by $\mathsf{REC}(\Sigma_G, \mathsf{PVar})$, is given by BNF syntax:

$$P ::= X \mid f(P_1, ..., P_l) \mid \mathrm{fix}(X = P),$$

where $X \in \mathsf{PVar}$, f is an operation symbol in Σ_G of arity l, and fix is a binding construct. This gives rise to the usual notions of free and bound variables in terms. The set of closed recursive terms (or *programs*) will be denoted by $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$. We shall assume a standard notion of substitution of terms for free process variables, and use, by abuse of notation, $P\{Q/X\}$ to denote term P in which each free occurrence of X has been replaced by Q, after possibly renaming bound variables in P. (The details of the operation of substitution in the presence of binders like fix are standard, and will not be important in this study. The interested reader is invited to consult, e.g., [88] for details). General substitutions mapping process variables to programs in $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$ will be denoted by boldface Greek letters like $\boldsymbol{\sigma}$.

We shall now define an operational semantics for the set of programs $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$ in terms of an lts with divergence, following the techniques presented in Section 3. Again, our aim is to give an operational semantics for recursive terms which is, as much as possible, in the spirit of the standard GSOS approach. One of the corner-stones of the GSOS philosophy is the use of structural recursion on terms to define the transition relation associated with a GSOS system. (Note that, for recursive terms, this requires defining the transition relation for arbitrary open terms in $\mathsf{REC}(\Sigma_G, \mathsf{PVar})$.) This suggests the following rule schemata to give the operational semantics of recursion:

$$\frac{x \xrightarrow{a} y}{\mathrm{fix}(X = x) \xrightarrow{a} y\{\mathrm{fix}(X = x)/X\}} \tag{2}$$

Such a variation on the rule for recursion used in, e.g., CCS [60] has been applied in, e.g., [28], and is discussed in detail in [13]. As argued in [13], for guarded recursive terms,[4] this rule for recursion gives the same results as the standard one based on unfoldings, namely:

$$\frac{x\{\mathrm{fix}(X = x)/X\} \xrightarrow{a} y}{\mathrm{fix}(X = x) \xrightarrow{a} y} \tag{3}$$

Moreover, rule (2) has the benefit of leading to an effective operational semantics that associates a finitely branching lts with each program [13]. However, in this paper, our main *desideratum* of a recursion rule is that it allows us to interpret recursive terms as fixed-points, i.e., we should like the following equation to hold:

$$\mathrm{fix}(X = P) = P\{\mathrm{fix}(X = P)/X\}. \tag{4}$$

This requirement rules out the use of rule (2), as, in general, (4) does not hold for unguarded recursive terms if their semantics is given using it. As an example, consider the GSOS system obtained by adding to $\mathsf{FINTREE}_\Omega$ the operation f given by the rule

$$\frac{x \xrightarrow{a} y}{f(x) \xrightarrow{b} \delta} \tag{5}$$

Then, as process variables have no transitions, the open term $f(X) + a.\delta$ can only exhibit one transition, namely

[4] In Milner's CCS, a program P is guarded if every occurence of a process variabler in P is within the scope of an $a.\_$ operation. Generalizations of this idea are presented in [6, 13, 91].

$f(X)+a.\delta \xrightarrow{a} \delta$. Therefore. the only transition that can be inferred for the term $P \equiv \mathrm{fix}(X=f(X)+a.\delta)$ using rule (2) is

$$\mathrm{fix}(X=f(X)+a.\delta) \xrightarrow{a} \delta. \tag{6}$$

On the other hand, the term $f(P)+a.\delta$, obtained by unwinding the recursive definition of P once, also has the possibility of performing a b-transition, which can be derived from rule (5) using transition (6). This implies that $P \not\sim f(P)+a.\delta$. For this reason, the semantics of recursive terms will be given in this study by means of the standard recursion rule (3).

In order to define the operational semantics of $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$, we need, first of all, to extend the convergence predicate to $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$. This can be done in standard fashion following, e.g., [8, 39, 46].

DEFINITION 4.1. The convergence predicate $\downarrow_{Grec}$ (abbreviated to $\downarrow$ when the GSOS system G is clear from the context) is the least predicate over $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$ that satisfies the following clauses:

1. $f(P_1, ..., P_l)\downarrow_{Grec}$ if
   (a) $f \neq \Omega$, and
   (b) for every argument i of f, if f tests i then $P_i\downarrow_{Grec}$.
2. $\mathrm{fix}(X=P)\downarrow_{Grec}$ if $P\{\mathrm{fix}(X=P)/X\}\downarrow_{Grec}$.

Again, we write $P\uparrow_{Grec}$ iff it is not the case that $P\downarrow_{Grec}$.

The motivation for the above definition is the following: a term P is divergent if its initial transitions are not fully specified. This occurs either when the initial behaviour of term P depends on underspecified arguments such as Ω or in the presence of unguarded recursive definitions. For example, the terms $\mathrm{fix}(X=X)$ and $\mathrm{fix}(X=f(X))$, where f is the operation given by rule (5), are *not* convergent as the initial behaviour of these processes depends on itself. It is immediate to see that the predicates $\downarrow_G$ and $\downarrow_{Grec}$ coincide over $\mathrm{T}(\Sigma_G)$, the set of recursion-free terms in $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$. We remark here that, when applied to SCCS [63] and the version of CCS considered in [94], the above definition delivers exactly the convergence predicates given by Hennessy in [39] and Walker in [94], respectively.

We shall now show how to associate a transition relation with a GSOS language with recursion. Of course, we should like the transition relation to be at least sound and supported in the sense of Definition 3.6.[5] Moreover, as it is the case for languages defined by rules without negative premises, we should like to associate the *least* such relation with a GSOS language with recursion. We shall now show that this can indeed be done, and that the extra structure given by the convergence predicate can be put to good use in giving a simple way of constructing the transition relation determined by a GSOS language with recursion. (The interested reader may wish to compare what follows with the techniques presented in the beautiful study [25].) The basic idea of the construction given in the proof of the following proposition is to build the transition relation associated with $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$ in two steps. In the first step of the construction, we derive the transitions emanating from *convergent terms* by induction on the convergence predicate. This we do by following the approach outlined in the previous section, and using the rule schemata (3) to derive the transitions of recursive terms. In the second, we use the information about the transitions that are possible for convergent terms to determine the outgoing transitions for all the terms in $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$.

The second step of the construction outlined informally above will use a notion of proof of a transition from a set of inference rules which is a slight modification of the standard one presented in, e.g., [37]. This we now present for the sake of completeness.

DEFINITION 4.2. Let G be a GSOS system. An *oracle* for G is a set O of transitions of the form $P \xrightarrow{a} Q$, where $P, Q \in \mathsf{CREC}(\Sigma_G, \mathsf{PVar})$ and $P\downarrow_{Grec}$.

Let O be an oracle for G. Let $P, Q \in \mathsf{CREC}(\Sigma_G, \mathsf{PVar})$ and $a \in \mathsf{Act}$. A proof with oracle O of the transition formula $P \xrightarrow{a} Q$ from G is a well-founded, upwardly branching tree whose nodes are labelled by positive transition formulae of the form $P' \xrightarrow{b} Q'$ with $P', Q' \in \mathsf{CREC}(\Sigma_G, \mathsf{PVar})$ and $b \in \mathsf{Act}$, such that:

- the root is labelled with $P \xrightarrow{a} Q$,
- if $P' \xrightarrow{c} Q'$ is the label of a node, and $\mathsf{Children}$ is the set of labels of the nodes directly above it, then:

— either there are an instance ψ/φ of rule (3) and a substitution $\sigma: \mathsf{Var} \to \mathsf{CREC}(\Sigma_G, \mathsf{PVar})$ such that $\{\psi\sigma\} = \mathsf{Children}$ and $\varphi\sigma = P' \xrightarrow{c} Q'$,

— or $\mathsf{Children} = \{P_i \xrightarrow{a_{ij}} Q_{ij} \mid 1 \leqslant i \leqslant l, 1 \leqslant j \leqslant m_i\}$, and there are a rule $r \in R_G$ of the form (1) and a substitution $\sigma: \mathsf{Var} \to \mathsf{CREC}(\Sigma_G, \mathsf{PVar})$ such that:

* $\mathsf{Children} = \bigcup_{i=1}^{l} \{\sigma(x_i) \xrightarrow{a_{ij}} \sigma(y_{ij}) \mid 1 \leqslant j \leqslant m_i\}$;

* for every $1 \leqslant i \leqslant l$ with $n_i > 0$, $\sigma(x_i)\downarrow_{Grec}$, and $(\sigma(x_i) \xrightarrow{b_{ik}} R) \in O$ $(1 \leqslant k \leqslant n_i)$ for no $R \in \mathsf{CREC}(\Sigma_G, \mathsf{PVar})$; and

* $P' \xrightarrow{c} Q' = (f(\mathbf{x}) \xrightarrow{c} C[\mathbf{x}, \mathbf{y}])\,\sigma$.

PROPOSITION 4.3. *For every GSOS language with recursion* $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$, *there exists a smallest sound and supported transition relation over* $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$. *This*

[5] To be precise, we should extend Definition 3.6 to rules involving arbitrary transition formulae. This is straightforward in the light of Definition 3.5. See also [25].

transition relation will be denoted by $\rightarrow_{Grec}$. *Moreover, for every* $P \in T(\Sigma_G)$,

$$P \xrightarrow{a}_{Grec} Q \Leftrightarrow Q \in T(\Sigma_G) \quad \textit{and} \quad P \xrightarrow{a}_{G} Q. \tag{7}$$

Proof. First of all, we show how to construct a sound and supported transition relation $\rightarrow_{Grec}$ over $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$. We then proceed to prove that $\rightarrow_{Grec}$ is indeed the smallest relation with these properties.

• *Construction of* $\rightarrow_{Grec}$. We shall construct the relation $\rightarrow_{Grec}$ over $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$ in two steps, and the construction outlined below will guarantee that the result is sound and supported. In the first step of the construction, we derive the transitions emanating from *convergent terms only*. In the second step, we use the information about the transitions that are possible for convergent terms to determine the outgoing transitions for all the terms in $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$.

— *Step* 1. We determine the transitions that are possible for convergent terms by induction on the predicate $\downarrow_{Grec}$. To this end, let us assume that $P\downarrow_{Grec}$. We shall now show how to construct the set

$$\mathsf{Der}(P) = \{(a, Q) \mid P \xrightarrow{a}_{Grec} Q\}$$

by examining the possible forms P may take.

∗ *Case* $P \equiv f(\mathbf{P})$. As $P\downarrow_{Grec}$, it must be the case that $f \neq \Omega$ and, for every argument i that is tested by f, $P_i\downarrow_{Grec}$. By the inductive hypothesis, we may then assume that we have already constructed the set $\mathsf{Der}(P_i)$ for each such i. Note that this means that we have complete information about the transitions that are possible from every argument tested by any rule for f. We now stipulate that $(a, Q) \in \mathsf{Der}(P)$ iff there exist a rule $r \in R_G$ for f of the form (1), and a substitution σ such that:

1. $\sigma(x_i) = P_i$ for every i,
2. $C[\mathbf{x}, \mathbf{y}]\,\sigma \equiv Q$,
3. for every positive transition formula $x_i \xrightarrow{a_{ij}} y_{ij}$ in H, we have that $(a_{ij}, \sigma(y_{ij})) \in \mathsf{Der}(P_i)$, and
4. for every negative transition formula $x_i \overset{b_{ik}}{\not\longrightarrow}$ in H, we have that for no R, $(b_{ik}, R) \in \mathsf{Der}(P_i)$.

∗ *Case* $P \equiv \mathrm{fix}(X = R)$. As $P\downarrow_{Grec}$, it must be the case that

$$R\{\mathrm{fix}(X = R)/X\} \downarrow_{Grec}.$$

By the inductive hypothesis, we may assume that we have already constructed the set $\mathsf{Der}(R\{\mathrm{fix}(X = R)/X\})$. We now stipulate that

$$\mathsf{Der}(P) = \mathsf{Der}(R\{\mathrm{fix}(X = R)/X\}).$$

It is immediate that, by construction, the transition relation for convergent terms defined above is indeed sound and supported. Moreover, it assigns a transition to a convergent recursion-free term iff $\rightarrow_G$ does.

— *Step* 2. Let $O = \{P \xrightarrow{a}_{Grec} Q \mid P\downarrow_{Grec} \text{ and } (a, Q) \in \mathsf{Der}(P)\}$. The transition relation is extended to divergent terms as follows: For a divergent term P the transition $P \xrightarrow{a}_{Grec} Q$ holds iff it has a proof with oracle O from G in the sense of Definition 4.2.

Again, the soundness and supportedness of the resulting transition relation is immediate by construction, and so is the agreement with $\rightarrow_G$ for recursion-free terms.

• *Leastness of* $\rightarrow_{Grec}$. First of all, it is easy to show, by induction on the convergence predicate, that any sound and supported transition relation $\Rightarrow$ must coincide with $\rightarrow_{Grec}$ for convergent processes, i.e., that for every convergent P, action a and $Q \in \mathsf{CREC}(\Sigma_G, \mathsf{PVar})$,

$$P \xrightarrow{a}_{Grec} Q \Leftrightarrow P \overset{a}{\Rightarrow} Q.$$

The leastness of $\rightarrow_{Grec}$ is then ensured by Step 2 of the above construction, and can be proven by a simple induction on the depth of the proof of transitions. ▮

To exemplify the construction in the above proof on a pathological example, let us consider the term $\mathrm{fix}(X = odd(X))$, where the operation *odd* is given by the rule

$$\frac{x \overset{a}{\not\longrightarrow}}{odd(x) \xrightarrow{a} \delta}$$

This operation is standardly used in the literature to show that negative premises and unguarded recursive definitions can lead to inconsistent specifications. (See, e.g., [20].) The reason for this phenomenon is that, if we follow the standard GSOS approach, the equation

$$X = odd(X) \tag{8}$$

can*not* have any solution. In fact, with the standard operational interpretation of GSOS systems and general transition system specifications with negative premises [25, 36], a process P solving the above equation would have to exhibit an a-transition iff it does not have one. In our approach, instead, the above equation has a unique solution modulo $\sim$. To see that this is indeed the case, note, first of all, that $\mathrm{fix}(X = odd(X))$ is a divergent term. It is then easy to see that, because of our requirement that negative premises in rules be interpreted over convergent terms only, the above rule can never be applied to derive a transition for $\mathrm{fix}(X = odd(X))$. Thus we have that this term has *no* transition and is divergent, i.e., that $\mathrm{fix}(X = odd(X)) \sim \Omega$. It is

easy to see that, modulo $\sim$, Ω is the unique solution of Eq. (8).

When applied to the languages considered in [39, 94], the theory that we have so far presented delivers exactly the transition system semantics for SCCS and CCS given in those references.

With the above definitions, the operational semantics of a GSOS language with recursion $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$ is given by the lts with divergence

$$\mathsf{lts}(G_{rec}) = (\mathsf{CREC}(\Sigma_G, \mathsf{PVar}), \mathsf{Act}, \rightarrow_{Grec}, \uparrow_{Grec}).$$

This lts in general is neither finitely branching nor image finite [44]. For example, the term $\mathrm{fix}(X = a.\delta \| X)$ in our running example has a countably infinite set of a-derivatives. However, the lts associated with a GSOS system with recursion is guaranteed to be *weakly finitely branching* in the sense of [1]. This is the import of the following result:

PROPOSITION 4.4 (Weak Finite branching). *Let G be a GSOS system. Then, for every $P \in \mathsf{CREC}(\Sigma_G, \mathsf{PVar})$, $P\downarrow_{Grec}$ implies that $\mathsf{Der}(P)$ is finite.*

Proof. By induction on the relation $\downarrow_{Grec}$. ∎

FACT 4.5. *Let G be a GSOS system. Then Eq. (4) is sound with respect to $\lesssim_{Grec}$, i.e., for every $P \in \mathsf{REC}(\Sigma_G, \mathsf{PVar})$ containing at most X free,*

$$\mathrm{fix}(X = P) \sim_{Grec} P\{\mathrm{fix}(X = P)/X\}.$$

Because Act is assumed to be finite, the lts with divergence giving semantics to a GSOS system with recursion is *a fortiori* sort-finite. As a corollary of general results by Abramsky, we then have the following characterization of the finitary bisimulation preorder over $\mathsf{lts}(G_{rec})$ for any GSOS language G:

PROPOSITION 4.6. *Let G be a GSOS system. Then the preorders $\lesssim^{\mathrm{F}}$ and $\lesssim_{\omega}$ coincide over $\mathsf{lts}(G_{rec})$.*

Proof. By Proposition 4.4, the lts $(\mathsf{CREC}(\Sigma_G, \mathsf{PVar}), \mathsf{Act}, \rightarrow_{Grec}, \uparrow_{Grec})$ is weakly finitely branching. By [1, Proposition 5.23], any weakly finitely branching lts satisfies Abramsky's axiom scheme of bounded non-determinacy (BN) (cf. [1, p. 193]). The claim then follows immediately by [1, Proposition 6.13], as our lts is sort-finite. ∎

We end this section with a result showing that prebisimulation is preserved by recursion.

DEFINITION 4.7. Let $P, Q \in \mathsf{REC}(\Sigma_G, \mathsf{PVar})$ contain at most $X_1, ..., X_n$ as free variables. Then $P \lesssim_{Grec} Q$ iff, for every vector of programs $R_1, ..., R_n \in \mathsf{CREC}(\Sigma_G, \mathsf{PVar})$,

$$P\{R_1/X_1, ..., R_n/X_n\} \lesssim_{Grec} Q\{R_1/X_1, ..., R_n/X_n\}.$$

THEOREM 4.8. *Let G be a GSOS system. Let $P, Q \in \mathsf{REC}(\Sigma_G, \mathsf{PVar})$ contain at most X as a free variable. Then $P \lesssim_{Grec} Q$ implies that $\mathrm{fix}(X = P) \lesssim_{Grec} \mathrm{fix}(X = Q)$.*

Proof. A generalization of the proof of [65, Proposition 4.12]. The details of the proof may be found in [9]. ∎

By Theorem 4.8 and an easy adaptation of the proof of Theorem 3.9 to $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$, we then have that:

COROLLARY 4.9. *Let G be a GSOS system. Then $\lesssim_{Grec}$ is a precongruence over the language $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$.*

To sum up our achievements, so far we have presented a novel operational treatment of GSOS languages that allows us to combine the use of rules with negative premises with arbitrary recursive definitions of behaviours. We have also shown that this operational view of processes induces operations that are well behaved with respect to prebisimulation, and we have proved a general characterization of the finitary part of the bisimulation preorder over arbitrary GSOS languages.

We now move on from the world of operational semantics to that of denotational semantics. In particular, we shall show how to construct denotational models for a class of GSOS languages with recursion that are guaranteed to be in complete agreement with the behavioural view of these languages presented so far.

5. BACKGROUND ON DENOTATIONAL SEMANTICS

In this section, we review the basic notions of algebraic semantics and domain theory that will be needed in the remainder of this study. The interested reader is invited to consult, e.g., [1, 30, 34, 38, 42, 69, 71, 86] for more details and extensive motivation.

5.1. *Preliminaries on Algebraic Semantics*

We assume that the reader is familiar with the basic notions of ordered and continuous algebras (see, e.g., [38, 42, 48]); however, in what follows we give a quick overview of the way a denotational semantics can be given to a recursive language like $\mathsf{REC}(\Sigma_G, \mathsf{PVar})$ following the standard lines of algebraic semantics [38]. The interested reader is invited to consult [42] for an explanation of the theory.

In what follows, we let Σ denote a signature in the sense of Section 3. A Σ-algebra is a pair $(\mathscr{A}, \Sigma_{\mathscr{A}})$ where $\mathscr{A}$ is the carrier set and $\Sigma_{\mathscr{A}}$ is a set of *operators* $f_{\mathscr{A}}: \mathscr{A}^l \to \mathscr{A}$, where $f \in \Sigma$ and $l = arity(f)$. We call $f_{\mathscr{A}}$ the *interpretation* of the function symbol f in the structure $\mathscr{A}$. Let $(\mathscr{A}, \Sigma_{\mathscr{A}})$ and $(\mathscr{B}, \Sigma_{\mathscr{B}})$ be Σ-algebras. A mapping $\varphi: \mathscr{A} \to \mathscr{B}$ is a *Σ-homomorphism* if it preserves the Σ-structure, i.e., if for every $f \in \Sigma$ and vector $\mathbf{a}$ of elements of $\mathscr{A}$ of the appropriate length,

$$\varphi(f_{\mathscr{A}}(\mathbf{a})) = f_{\mathscr{B}}(\varphi(\mathbf{a})).$$

The *term algebra* $\mathrm{T}(\Sigma)$ is the *initial* Σ-algebra; i.e., if $(\mathscr{A}, \Sigma_{\mathscr{A}})$ is a Σ-algebra then there is a unique Σ-homomorphism $\iota_{\mathscr{A}}: \mathrm{T}(\Sigma) \to \mathscr{A}$. We refer to this homomorphism as the *interpretation* of $\mathrm{T}(\Sigma)$ in $\mathscr{A}$.

A *Σ-domain* $(\mathscr{A}, \sqsubseteq_{\mathscr{A}}, \Sigma_{\mathscr{A}})$ is a Σ-algebra whose carrier $(\mathscr{A}, \sqsubseteq_{\mathscr{A}})$ is an algebraic complete partial order (cpo) (see, e.g., [71]) and whose operations are interpreted as continuous functions. The notion of *Σ-poset* (respectively *Σ-preorder*) may be defined in a similar way by requiring that $(\mathscr{A}, \sqsubseteq_{\mathscr{A}})$ be a partially ordered (resp. preordered) set and that the operators be monotonic. The notion of Σ-homomorphism extends to the ordered Σ-structures in the obvious way by requiring that such maps preserve the underlying order-theoretic structure as well as the Σ-structure.

For any Σ-structure $\mathscr{A}$, be it ordered or not, the set $[\mathsf{PVar} \to \mathscr{A}]$ of $\mathscr{A}$-environments will be denoted by $\mathrm{ENV}_{\mathscr{A}}$ and ranged over by the meta-variable ρ. The (unique) interpretation of $\mathbb{T}(\Sigma, \mathsf{PVar})$ in $\mathscr{A}$ is the mapping $\mathscr{A}[\![\cdot]\!]: \mathbb{T}(\Sigma, \mathsf{PVar}) \to [\mathrm{ENV}_{\mathscr{A}} \to \mathscr{A}]$ defined recursively by

$$\mathscr{A}[\![X]\!]\,\rho \triangleq \rho(X)$$
$$\mathscr{A}[\![f(P_1, ..., P_l)]\!]\,\rho \triangleq f_{\mathscr{A}}(\mathscr{A}[\![P_1]\!]\,\rho, ..., \mathscr{A}[\![P_l]\!]\,\rho).$$

If $\mathscr{A}$ is a Σ-domain the interpretation extends to the set $\mathsf{REC}(\Sigma, \mathsf{PVar})$ of recursive terms over Σ by setting

$$\mathscr{A}[\![\mathrm{fix}(X = P)]\!]\,\rho \triangleq \mathsf{Y}\lambda a.\mathscr{A}[\![P]\!]\,\rho[X \to a],$$

where Y denotes the least fixed-point operator. As usual, $\rho[X \to a]$ denotes the environment which is defined as follows:

$$\rho[X \to a](Y) \triangleq \begin{cases} a & \text{if } X = Y \\ \rho(Y) & \text{otherwise.} \end{cases}$$

Note that, for each closed recursive term $P \in \mathsf{CREC}(\Sigma, \mathsf{PVar})$, $\mathscr{A}[\![P]\!]\,\rho$ does not depend on the environment ρ. The denotation of a closed term P will be denoted by $\mathscr{A}[\![P]\!]$. For every closed, recursion-free term P, $\mathscr{A}[\![P]\!]$ coincides with $\iota_{\mathscr{A}}(P)$.

In what follows, we shall make use of some general results about the semantic mappings defined above, which may be found in [30, 38, 42]. The first states that for any $P \in \mathsf{REC}(\Sigma, \mathsf{PVar})$ there is a sequence of *finite approximations* $P^n \in \mathbb{T}(\Sigma, \mathsf{PVar})$ $(n \in \mathbb{N})$ such that, for any Σ-domain $\mathscr{A}$,

$$\mathscr{A}[\![P]\!] = \bigsqcup_{n \geqslant 0} \mathscr{A}[\![P^n]\!]. \tag{9}$$

The second states that there is a syntactically defined relation between P and every finite approximation P^n. Let $\leqslant_{\Omega}$ be the least Σ-precongruence which satisfies Eq. (4) and the inequality

$$\Omega \leqslant X. \tag{10}$$

Then, for every $n \geqslant 0$, $P^n \leqslant_{\Omega} P$. (Note that the relation $\leqslant_{\Omega}$ is contained in the behavioural preorders $\lesssim$ and $\lesssim_{\omega}$.)

For any binary relation $\mathfrak{R}$ over $\mathsf{CREC}(\Sigma, \mathsf{PVar})$, the algebraic part of $\mathfrak{R}$, denoted by $\mathfrak{R}^A$, is defined as follows [42]:

$$P\,\mathfrak{R}^A\,Q \Leftrightarrow \forall n\,\exists m.P^n\,\mathfrak{R}\,Q^m.$$

We say that $\mathfrak{R}$ is *algebraic* iff $\mathfrak{R} = \mathfrak{R}^A$. Intuitively, a relation is algebraic if it is completely determined by how it behaves on recursion-free terms. Every denotational interpretation $\mathscr{A}[\![\cdot]\!]$ induces a preorder $\sqsubseteq_{\mathscr{A}}$ over terms by

$$P \sqsubseteq_{\mathscr{A}} Q \Leftrightarrow \mathscr{A}[\![P]\!] \sqsubseteq_{\mathscr{A}} \mathscr{A}[\![Q]\!].$$

The following result characterizes a class of denotational interpretations which induce relations over terms that are algebraic.

LEMMA 5.1. *Let $\mathscr{A}$ be a Σ-domain. Then the following statements hold*:

1. *For all $P, Q \in \mathsf{CREC}(\Sigma, \mathsf{PVar})$, $P \sqsubseteq^A_{\mathscr{A}} Q$ implies $P \sqsubseteq_{\mathscr{A}} Q$.*
2. *Assume that, for every $P \in T(\Sigma)$, $\mathscr{A}[\![P]\!]$ is a compact element in $\mathscr{A}$. Then $\sqsubseteq_{\mathscr{A}}$ is algebraic.*

Proof. Let Σ be a signature, and let $\mathscr{A}$ be a Σ-domain. We prove the two statements separately.

1. Let $P, Q \in \mathsf{CREC}(\Sigma, \mathsf{PVar})$ and assume that $P \sqsubseteq^A_{\mathscr{A}} Q$ We prove that $P \sqsubseteq_{\mathscr{A}} Q$ as follows:

$$\begin{aligned}
P \sqsubseteq^A_{\mathscr{A}} Q &\Leftrightarrow \forall n \geqslant 0\ \exists m \geqslant 0 : P^n \sqsubseteq_{\mathscr{A}} Q^m \\
&\qquad \text{(By the definition of } \sqsubseteq^A_{\mathscr{A}}\text{)} \\
&\Leftrightarrow \forall n \geqslant 0\ \exists m \geqslant 0 : \mathscr{A}[\![P^n]\!] \sqsubseteq_{\mathscr{A}} \mathscr{A}[\![Q^m]\!] \\
&\qquad \text{(By the definition of } \sqsubseteq_{\mathscr{A}}\text{)} \\
&\Rightarrow \forall n \geqslant 0 : \mathscr{A}[\![P^n]\!] \sqsubseteq_{\mathscr{A}} \bigsqcup_{m \geqslant 0} \mathscr{A}[\![Q^m]\!] \\
&\qquad (\star) \\
&\Leftrightarrow \bigsqcup_{n \geqslant 0} \mathscr{A}[\![P^n]\!] \sqsubseteq_{\mathscr{A}} \bigsqcup_{m \geqslant 0} \mathscr{A}[\![Q^m]\!] \\
&\Leftrightarrow \mathscr{A}[\![P]\!] \sqsubseteq_{\mathscr{A}} \mathscr{A}[\![Q]\!] \\
&\qquad \text{(By Eq. (9))} \\
&\Leftrightarrow P \sqsubseteq_{\mathscr{A}} Q.
\end{aligned}$$

2. To prove this statement we note that if $\mathscr{A}[\![P^n]\!]$ is a compact element in $\mathscr{A}$ then the implication labelled by ($\star$) in the proof above may be replaced by a bi-implication. ▮

In view of the above general lemma, the relations over terms induced by a denotational semantics are always algebraic, provided that the denotations of recursion-free terms are compact elements in the cpo $\mathscr{A}$. We shall make use of this fact in the technical developments to follow.

5.2. *A Domain Equation for Synchronization Trees*

In this section we recall Abramsky's domain equation for synchronization trees, and introduce the background in domain theory that is necessary to understand the remainder of the paper. The interested reader is referred to, e.g., [38, 42, 71, 86, 87] for more general information on the theory of domains and denotational semantics, and to [1, Section 3] for a quick reference to some of the results mentioned in this section.

The canonical domain we shall use to give a denotational semantics to a class of GSOS languages is the domain of synchronization trees over a countable set of labels Lab considered by Abramsky in his seminal paper [1]. This is defined to be the initial solution $\mathscr{D}(\mathsf{Lab})$ in the category **SFP** (cf. [69]) of the domain equation

$$D = (\mathbf{1})_\perp \oplus P\left[\sum_{l \in \mathsf{Lab}} D\right], \tag{11}$$

where $\mathbf{1}$ is the one point domain, $(\cdot)_\perp$ is lifting, $\oplus$ is coalesced sum, $\sum$ is separated sum, and $P[D]$ denotes the Plotkin powerdomain of D (cf. [69, 71] for details on these domain-theoretic operations). We henceforth omit the parameter Lab as it will be always clear from the context.

To streamline the presentation and make our results more accessible to uninitiated readers, in this study we shall abstract completely from the domain-theoretic description of $\mathscr{D}$ given by (11). Our description of the domain of synchronization trees $\mathscr{D}$ will follow the one given in [48], and we shall rely on results presented in that reference that show how to construct $\mathscr{D}$ starting from a suitable preorder on the set of finite synchronization trees $\mathsf{ST}(\mathsf{Lab})$. Our reconstruction of $\mathscr{D}$ will be given in three steps:

1. First of all, we shall define a preorder $\sqsubseteq$ on the set of finite synchronization trees $\mathsf{ST}(\mathsf{Lab})$. This preorder will be a reformulation of the Egli–Milner preorder over $\mathsf{ST}(\mathsf{Lab})$ presented in [48]. (See Fact 5.5 below.)

2. Second, we shall relate the poset of compact elements of $\mathscr{D}$ to the poset of equivalence classes induced by the preorder $(\mathsf{ST}(\mathsf{Lab}), \sqsubseteq)$.

3. Finally, we shall use the fact that $\mathscr{D}$ is the ideal completion of its poset of compact elements to relate it to $(\mathsf{ST}(\mathsf{Lab}), \sqsubseteq)$.

The approach outlined above will allow us to factor the definition of the continuous algebra structure [34, 38, 42] on $\mathscr{D}$ given in Section 5 in three similar steps, hopefully making it simpler to understand.

DEFINITION 5.2. We define $\sqsubseteq$ as the least binary relation over $\mathsf{ST}(\mathsf{Lab})$ satisfying:

$t \sqsubseteq u$ if (1) $\langle l, t' \rangle \in t \Rightarrow \exists \langle l, u' \rangle \in u : t' \sqsubseteq u'$ and

(2) $\perp \in u \Rightarrow \perp \in t$ and

(3) $\langle l, u' \rangle \in u \Rightarrow (\perp \in t \text{ or } \exists \langle l, t' \rangle \in t : t' \sqsubseteq u')$.

The relation $\sqsubseteq$ so defined is easily seen to be a preorder over $\mathsf{ST}(\mathsf{Lab})$, whose kernel will be denoted by $\simeq$. Moreover, it has the following useful property:

FACT 5.3. *For all $t, u \in \mathsf{ST}(\mathsf{Lab})$, $t \sqsubseteq u$ iff $t \lesssim u$.*

Proof. The "only if" implication follows because $\lesssim$ satisfies the defining constraints of $\sqsubseteq$, and $\sqsubseteq$ is the smallest such relation. The proof of the "if" implication is an easy induction on the combined size of the trees t and u. ▮

DEFINITION 5.4. $(\mathsf{ST}(\mathsf{Lab})/\simeq, \sqsubseteq_\simeq)$ stands for the poset whose elements are $\simeq$-equivalence classes of synchronization trees (denoted by $[t]$), and whose partial ordering $\sqsubseteq_\simeq$ is given by

$$[t] \sqsubseteq_\simeq [u] \Leftrightarrow t \sqsubseteq u.$$

We can now relate the preorder of synchronization trees $(\mathsf{ST}(\mathsf{Lab}), \sqsubseteq)$ with the poset of compact elements of $\mathscr{D}$ in a way that will allow us to defines, in a canonical way, continuous operations on $\mathscr{D}$ from monotonic ones on $(\mathsf{ST}(\mathsf{Lab}), \sqsubseteq)$.

First of all, we recall from [1] that $\mathscr{D}$ is, up to isomorphism, the algebraic complete partial order (cpo) whose poset of compact elements $(\mathscr{K}(\mathscr{D}), \sqsubseteq_{\mathscr{K}(\mathscr{D})})$ is given in [1] by:

- $\mathscr{K}(\mathscr{D})$ is defined inductively as follows:
  - $\varnothing \in \mathscr{K}(\mathscr{D})$
  - $\{\perp\} \in \mathscr{K}(\mathscr{D})$
  - $l \in \mathsf{Lab}, d \in \mathscr{K}(\mathscr{D}) \Rightarrow \{\langle l, d \rangle\} \in \mathscr{K}(\mathscr{D})$
  - $d_1, d_2 \in \mathscr{K}(\mathscr{D}) \Rightarrow \mathsf{Con}(d_1 \cup d_2) \in \mathscr{K}(\mathscr{D})$, where Con denotes the convex closure operation (see, e.g., [1, p. 170]);
- $\sqsubseteq_{\mathscr{K}(\mathscr{D})}$ is defined by

$$d \sqsubseteq_{\mathscr{K}(\mathscr{D})} e \Leftrightarrow d = \{\perp\} \quad \text{or} \quad d \sqsubseteq_{\mathrm{EM}} e$$

where $\sqsubseteq_{\mathrm{EM}}$ denotes the standard Egli–Milner ordering (see, e.g., [1, Definition 3.3]).

From the above definitions, it follows that $\mathscr{K}(\mathscr{D})$ is a subset of the set of finite synchronization trees $\mathsf{ST}(\mathsf{Lab})$. Hence it makes sense to compare the relations $\sqsubseteq$ and $\sqsubseteq_{\mathscr{K}(\mathscr{D})}$ over it. The following small result lends credence to our previous claims:

FACT 5.5. *For all $d, e \in \mathscr{K}(\mathscr{D})$, $d \sqsubseteq e$ iff $d \sqsubseteq_{\mathscr{K}(\mathscr{D})} e$.*

Proof. From Abramsky's results (see [1, Proposition 3.11]), we have that $\mathscr{D}$ is "internally fully abstract," i.e., that for all $d_1, d_2 \in \mathscr{D}$,

$$d_1 \lesssim d_2 \Leftrightarrow d_1 \sqsubseteq_{\mathscr{D}} d_2.$$

The result now follows from Fact 5.3 and the fact that each compact element of $\mathscr{D}$ is in $\mathsf{ST}(\mathsf{Lab})$. ∎

As a consequence of this result, to ease the presentation of the technical results to follow, from now on we shall always use $\sqsubseteq$ as our notion of preorder on $\mathscr{K}(\mathscr{D})$. Using it, we may rephrase the definition of convex-closure of a synchronization tree as follows:

FACT 5.6. *Let $t = \{\langle l_i, t_i\rangle \mid 1 \leqslant i \leqslant n\}[\cup\{\bot\}]$ be a synchronization tree in $\mathsf{ST}(\mathsf{Lab})$. Then its convex-closure $\mathsf{Con}(t)$ is given by*:

- $\bot \in \mathsf{Con}(t)$ *iff* $\bot \in t$;
- $\langle l, t'\rangle \in \mathsf{Con}(t)$ *iff one of the following holds*:

— *there exist $\langle l_i, t_i\rangle, \langle l_j, t_j\rangle \in t$ such that $l_i = l_j = l$ and $t_i \sqsubseteq t' \sqsubseteq t_j$; or*

— *$\bot \in t$ and, for some $\langle l_i, t_i\rangle \in t$, $l_i = l$ and $t' \sqsubseteq t_i$.*

For a synchronization tree $t = \{\langle l_i, t_i\rangle \mid 1 \leqslant i \leqslant n\}[\cup\{\bot\}]$, its recursive convex-closure t^{c} is inductively defined as follows:

$$t^{\mathrm{c}} \triangleq \mathsf{Con}(\{\langle l_i, t_i^{\mathrm{c}}\rangle \mid 1 \leqslant i \leqslant n\}[\cup\{\bot\}]). \tag{12}$$

It is not difficult to see that, for every $t \in \mathsf{ST}(\mathsf{Lab})$, t^{c} is a compact element of $\mathscr{D}$. Moreover, the function $(\cdot)^{\mathrm{c}}: (\mathsf{ST}(\mathsf{Lab}), \sqsubseteq) \to (\mathscr{K}(\mathscr{D}), \sqsubseteq)$ enjoys the following properties:

LEMMA 5.7.

1. *For every $t \in \mathsf{ST}(\mathsf{Lab})$, $t \simeq t^{\mathrm{c}}$.*
2. *For all $t, u \in \mathsf{ST}(\mathsf{Lab})$, $t \sqsubseteq u$ iff $t^{\mathrm{c}} \sqsubseteq u^{\mathrm{c}}$.*
3. *For all $t, u \in \mathsf{ST}(\mathsf{Lab})$, $t \simeq u$ iff $t^{\mathrm{c}} = u^{\mathrm{c}}$, i.e., t^{c} and u^{c} are equal as sets.*
4. *For all $t \in \mathsf{ST}(\mathsf{Lab})$, $t^{\mathrm{c}} = (t^{\mathrm{c}})^{\mathrm{c}}$.*

As an immediate consequence of the above lemma we have the following:

COROLLARY 5.8. *The poset $(\mathsf{ST}(\mathsf{Lab})/\simeq, \sqsubseteq_{\simeq})$ is isomorphic to $(\mathscr{K}(\mathscr{D}), \sqsubseteq_{\mathscr{K}(\mathscr{D})})$ under the isomorphism given by $\varphi([t]) = t^{\mathrm{c}}$.*

Assume now that $\mathbf{f}_{\mathsf{ST}}: (\mathsf{ST}(\mathsf{Lab}), \sqsubseteq)^I \to (\mathsf{ST}(\mathsf{Lab}), \sqsubseteq)$ is a monotonic function. We may naturally use $\mathbf{f}_{\mathsf{ST}}$ to define a function $\mathbf{f}_{\mathscr{K}(\mathscr{D})}: (\mathscr{K}(\mathscr{D}), \sqsubseteq)^I \to (\mathscr{K}(\mathscr{D}), \sqsubseteq)$ as follows:

$$\mathbf{f}_{\mathscr{K}(\mathscr{D})}(\mathbf{t}^{\mathrm{c}}) \triangleq (\mathbf{f}_{\mathsf{ST}}(\mathbf{t}))^{\mathrm{c}}. \tag{13}$$

It is easy to see that the function is well defined in this way. To see that it is monotonic, assume that $\mathbf{t}^{\mathrm{c}} \sqsubseteq \mathbf{u}^{\mathrm{c}}$. Then:

$$\begin{aligned}
\mathbf{t}^{\mathrm{c}} \sqsubseteq \mathbf{u}^{\mathrm{c}} &\Leftrightarrow \mathbf{t} \sqsubseteq \mathbf{u} \\
&\quad \text{(Lemma 5.7(2))} \\
&\Rightarrow \mathbf{f}_{\mathsf{ST}}(\mathbf{t}) \sqsubseteq \mathbf{f}_{\mathsf{ST}}(\mathbf{u}) \\
&\quad (\mathbf{f}_{\mathsf{ST}} \text{ is monotonic}) \\
&\Leftrightarrow (\mathbf{f}_{\mathsf{ST}}(\mathbf{t}))^{\mathrm{c}} \sqsubseteq (\mathbf{f}_{\mathsf{ST}}(\mathbf{u}))^{\mathrm{c}} \\
&\quad \text{(Lemma 5.7(2))} \\
&\Leftrightarrow \mathbf{f}_{\mathscr{K}(\mathscr{D})}(\mathbf{t}^{\mathrm{c}}) \sqsubseteq \mathbf{f}_{\mathscr{K}(\mathscr{D})}(\mathbf{u}^{\mathrm{c}}) \\
&\quad (13)
\end{aligned}$$

In what follows we refer to the function $\mathbf{f}_{\mathscr{K}(\mathscr{D})}$ defined above as $\mathbf{f}^{\mathrm{c}}_{\mathsf{ST}}$. We extend this notation to a set of operators $\Sigma_{\mathsf{ST}(\mathsf{Lab})}$ over synchronization trees in the standard way, i.e., $\Sigma^{\mathrm{c}}_{\mathsf{ST}(\mathsf{Lab})} = \{\mathbf{f}^{\mathrm{c}}_{\mathsf{ST}} \mid \mathbf{f}_{\mathsf{ST}} \in \Sigma_{\mathsf{ST}(\mathsf{Lab})}\}$.

An easy consequence of the previous theory is that, for any signature Σ, Eq. (13) can be used to induce a Σ-poset structure on the poset of compact elements from a Σ-preorder structure on finite synchronization trees. This is formalized in the following result, which also relates the unique meaning maps from $\mathrm{T}(\Sigma)$ to the resulting algebraic structures, denoted by $\mathsf{ST}[\![\cdot]\!]$ and $\mathscr{K}(\mathscr{D})[\![\cdot]\!]$ respectively.

COROLLARY 5.9. *For any signature Σ, if $(\mathsf{ST}(\mathsf{Lab}), \sqsubseteq, \Sigma_{\mathsf{ST}(\mathsf{Lab})})$ is a Σ-preorder then*:

1. *$(\mathscr{K}(\mathscr{D}), \sqsubseteq_{\mathscr{K}(\mathscr{D})}, \Sigma^{\mathrm{c}}_{\mathsf{ST}(\mathsf{Lab})})$ is a Σ-poset, and*
2. *for every $P \in T(\Sigma)$, $\mathscr{K}(\mathscr{D})[\![P]\!] = (\mathsf{ST}[\![P]\!])^{\mathrm{c}}$.*

Proof. The first statement follows immediately from the previous theory. The second follows from the initiality of $\mathrm{T}(\Sigma)$, because the mappings $\mathscr{K}(\mathscr{D})[\![\cdot]\!]$ and $(\cdot)^{\mathrm{c}} \circ \mathsf{ST}[\![\cdot]\!]$ are both Σ-homomorphisms. ∎

The corollary above implies that we can lift any Σ-preorder structure on $(\mathsf{ST}(\mathsf{Lab}), \sqsubseteq)$ to a Σ-poset structure on $(\mathscr{K}(\mathscr{D}), \sqsubseteq)$, in the sense of [42], in a canonical way. In Section 6.2 we shall take advantage of this fact. Finally, from the theory of powerdomains [48, 69, 83], we know that the domain of synchronization trees $\mathscr{D}$ is, up to isomorphism, the ideal completion of the poset of compact

elements $\mathscr{K}(\mathscr{D})$. As a result of this observation, we can extend any monotonic function $\mathbf{f}_{\mathscr{K}(\mathscr{D})}: (\mathscr{K}(\mathscr{D}), \sqsubseteq)^I \to (\mathscr{K}(\mathscr{D}), \sqsubseteq)$ to a continuous function $\mathbf{f}_{\mathscr{D}}: (\mathscr{D}, \sqsubseteq_{\mathscr{D}})^I \to (\mathscr{D}, \sqsubseteq_{\mathscr{D}})$ by:

$$\mathbf{f}_{\mathscr{D}}(\mathbf{k}) \triangleq \bigsqcup \{\mathbf{f}_{\mathscr{K}(\mathscr{D})}(\mathbf{d}) \mid \mathbf{d} \in \mathscr{K}(\mathscr{D}) \text{ and } \mathbf{d} \sqsubseteq_{\mathscr{D}} \mathbf{k}\}. \quad (14)$$

The interested reader is invited to consult, e.g., [42, Section 3.3] for a discussion of the properties afforded by this canonical extension. Thus (14) can be used to conservatively extend any Σ-poset algebra structure on $(\mathscr{K}(\mathscr{D}), \sqsubseteq)$ to a continuous algebra structure on $\mathscr{D}$, in the sense of [34, 38, 42].

6. DENOTATIONAL SEMANTICS FOR COMPACT GSOS LANGUAGES

In this section we shall present a general technique to give denotational semantics in terms of the Plotkin powerdomain of synchronization trees (see Section 5.2) for a class of GSOS languages with recursion. The denotational semantics will be guaranteed to be *fully abstract*, in the sense of Milner and Plotkin [57, 58, 70, 87], with respect to the finitary part of the prebisimulation relation. The languages that we shall consider have the structure of most standard process calculi (see, e.g., [12, 18, 47, 65]); they will consist of a set of operations to build finite, acyclic labelled transition systems and a facility for recursive definitions of behaviours. Thus we shall consider GSOS languages with recursion in which infinite behaviours can only be defined by means of recursive definitions.

6.1. *Compact GSOS Systems*

The following notion from [7] will allow us to pin down precisely a class of GSOS operations that map finite processes to finite processes. The semantic counterparts of these operations will have the property of being *compact* in the sense of [48], i.e., of mapping compact elements in the Plotkin powerdomain of synchronization trees to compact elements. In view of Lemma 5.1, denotational interpretations for the resulting languages will induce algebraic preorders over terms.

DEFINITION 6.1. A GSOS rule of the general form (1) is *linear* if each variable occurs at most once in the target and, for each argument i that is tested positively, x_i does not occur in the target and at most one of the y_{ij}'s does. An operation from a GSOS system G is linear iff all rules for it are linear. Finally, G itself is linear iff it only contains linear rules.

The format of linear rules is a restriction of the general GSOS format in that no copying of arguments is allowed and no argument for which there is a positive antecedent may appear in the target of a rule. Moreover, there may be many positive antecedents for an argument x_i in a rule, but at most one of the y_{ij}'s may appear in its target. As far as we know, all the operations occurring in the standard process algebras are linear. An example of a non-linear operation is the Kleene star operation [52] given by the rules (one pair of rules for each $a \in \mathsf{Act}$)

$$\frac{x \xrightarrow{a} x'}{x^*y \xrightarrow{a} x';(x^*y)} \qquad \frac{y \xrightarrow{a} y'}{x^*y \xrightarrow{a} y'}$$

Modulo a different treatment of termination, these rules may be found in [19], where the Kleene star operation is considered in the setting of ACP. Another example of a non-linear operation is the replication operation of the π-calculus [67] given by the rules (one rule for each $a \in \mathsf{Act}$)

$$\frac{x \xrightarrow{a} x'}{!x \xrightarrow{a} x' \mid !x}$$

The following syntactic constraint on GSOS systems from [7] will be used to characterize a class of such languages, the so-called syntactic well-founded ones, in which infinite behaviours can only be defined by means of recursive definitions. This is achieved by imposing a notion of weight on recursion-free terms that will be shown to be decreasing with transitions for linear GSOS languages. (See Proposition 6.4 below).

DEFINITION 6.2 [7]. A *weight function* for a GSOS system G is a mapping w from operation symbols in Σ_G to natural numbers. The extension of w to $\mathbb{T}(\Sigma_G, \mathsf{Var})$ is the function $W: \mathbb{T}(\Sigma_G) \to \mathbb{N}$ given by

$$W(x) \triangleq 0$$
$$W(f(P_1, ..., P_l)) \triangleq w(f) + w(P_1) + \cdots + W(P_l).$$

A GSOS system G is *syntactically well founded* iff there exists a weight function w such that, for each rule $r \in R_G$ with principal operation symbol f and target $C[\mathbf{x}, \mathbf{y}]$ the following conditions hold:

- if r has no positive antecedents then $W(C[\mathbf{x}, \mathbf{y}]) < w(f)$, and
- $W(C[\mathbf{x}, \mathbf{y}]) \leqslant w(f)$ otherwise.

For example, the GSOS system in Fig. 1 is linear and syntactically well founded. In fact, it is sufficient to assign weight 1 to the action prefixing operations and weight 0 to all the other operations. On the other hand, no GSOS system containing a constant a^ω with rule

$$\frac{}{a^\omega \xrightarrow{a} a^\omega} \quad (15)$$

can be syntactically well founded. Syntactic well-foundedness is decidable over GSOS systems (cf. [7, Theorem 6.8]), and, for linear GSOS systems, it is sufficient to guarantee that terms are semantically well founded in the sense of [7].

DEFINITION 6.3 (Compact GSOS Systems). A GSOS system is said to be *compact* iff it is linear and syntactically well founded.

An example of a compact GSOS system is our running example, the language $\mathrm{preACP}_{\Omega 0}$ in Fig. 1. Other examples are the finite alphabet versions of standard process algebras such as CCS [65], CSP [47], and ACP [18]. The following proposition, which can be proven following the lines of [7, Proposition 6.7], states the key property of compact GSOS systems, namely that no term in a compact GSOS system exhibits infinite derivations.

PROPOSITION 6.4. *Let G be a compact GSOS system. Then G is well founded, i.e., for every $P \in T(\Sigma_G)$ there exists no infinite sequence $P_0, a_0, P_1, a_1, P_2, \ldots$ of terms in $T(\Sigma_G)$ and actions in* Act *with $P \equiv P_0$ and $P_i \xrightarrow{a}_G P_{i+1}$ for all $i \geqslant 0$.*

As an immediate corollary of the above result and of Lemma 3.8, if G is a compact GSOS system, then we can unfold the finite, acyclic process graph giving the operational semantics of a term $P \in T(\Sigma_G)$ to obtain

COROLLARY 6.5. *Let G be a compact GSOS system. Then, for every $P \in T(\Sigma_G)$, there exists a synchronization tree $t_P \in \mathsf{ST}(\mathsf{Act})$ such that $P \sim t_P$.*

In what follows, we shall give a denotational semantics for GSOS systems with recursion built on top of compact GSOS systems. In view of Corollary 5.9, in order to endow $\mathscr{D}$ with a structure of a continuous algebra [30, 34, 38, 42], it is sufficient to define a monotonic Σ-structure on $\mathsf{ST}(\mathsf{Act})$. An interpretation for the GSOS language with recursion $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$ built on top of G can then be given in standard fashion, and will be shown to induce a fully abstract semantics for $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$ with respect to $\lesssim_\omega$.

6.2. *A Fully Abstract Denotational Semantics for Compact GSOS Systems*

We shall now present a general method to give a denotational semantics in terms of the Plotkin powerdomain of synchronization trees to compact GSOS systems. We shall then show how to extend the results of this section to GSOS languages with recursion built on top of such systems.

Let G be a compact GSOS language. We give a way of defining, for each Σ_G-context $C[\mathbf{x}]$, a function $\mathbf{C}_{\mathsf{ST}}$ over $\mathsf{ST}(\mathsf{Act})$ of the appropriate arity. The definition of $\mathbf{C}_{\mathsf{ST}}$ will be given using the rules in R_G as a guideline. First of all, note that it is sufficient to define semantic operations $\mathbf{f}_{\mathsf{ST}}$ for each $f \in \Sigma_G$, as derived semantic operations can then be obtained by function composition. The definition of the functions $\mathbf{f}_{\mathsf{ST}}$ is given by the inductive construction in Definition 6.9. Intuitively, the inductive construction of the synchronization tree $\mathbf{f}_{\mathsf{ST}}(\mathbf{t})$ given in Definition 6.9 is well founded because, by the compactness of G, whenever the premises of a rule of the form (1) can be met by a vector of finite synchronization trees $\mathbf{t}$ (viewed as a vector of lts's), then either the weight of $C[\mathbf{x}, \mathbf{y}]$ is strictly smaller than that of f, or the weight of $C[\mathbf{x}, \mathbf{y}]$ is the same as that of f, and the sum of the sizes of the arguments of $\mathbf{C}_{\mathsf{ST}}$ has decreased.

Before presenting the inductive definition of the synchronization tree $\mathbf{f}_{\mathsf{ST}}(\mathbf{t})$, we now put the intuitive justification of its well-foundedness on firmer ground. First of all, we associate with a class of relevant Σ_G-contexts of the form $C[\mathbf{x}]$ a measure of the complexity of the synchronization tree $\mathbf{C}_{\mathsf{ST}}[\mathbf{t}]$, where $\mathbf{t}$ is a vector of synchronization trees of the appropriate length.

DEFINITION 6.6. The *height* of a synchronization tree $t = \{\langle a_1, t_1\rangle, \ldots, \langle a_n, t_n\rangle\}[\cup\{\bot\}] \in \mathsf{ST}(\mathsf{Act})$ is the positive integer inductively defined by

$$\mathsf{ht}(\{\langle a_1, t_1, \rangle, \ldots, \langle a_n, t_j\rangle\})[\cup\{\bot\}] = \sup\{\mathsf{ht}(t_i) \mid 1 \leqslant i \leqslant n\} + 1.$$

By convention, the supremum of the empty set is 0.

DEFINITION 6.7. Let G be a compact GSOS system, and let $w: \Sigma_G \to \mathbb{N}$ be a weight function for G in the sense of Definition 6.2. For each Σ_G-contest $C[\mathbf{x}]$ in which each variable occurs at most once, and for each vector of synchronization trees $\mathbf{t}$ of the appropriate length we define the pair of natural numbers $\mathsf{norm}(C[\mathbf{x}], \mathbf{t})$ as

$$\mathsf{norm}(C[\mathbf{x}], \mathbf{t}) = (W(C[\mathbf{x}]), \textstyle\sum \{\mathsf{ht}(t_i) \mid x_i \text{ occurs in } C[\mathbf{x}]\}).$$

Following [7, Proposition 6.7], we can now state the following technical result, whose proof may be found in [9].

PROPOSITION 6.8. *Let G be a compact GSOS system. Then, for every rule r in R_G of the form* (1)*, and vectors of trees $\mathbf{t} = t_1, \ldots, t_l$, $\mathbf{u} = u_{11} \cdots u_{1m_1} \cdots u_{l1} \cdots u_{lm_l}$ such that, for all $1 \leqslant i \leqslant l$, $1 \leqslant j \leqslant m_i$, $\langle a_{ij}, u_{ij}\rangle \in t_i$,*

$$\mathsf{norm}(C[\mathbf{x}, \mathbf{y}], \mathbf{tu}) \prec \mathsf{norm}(f(\mathbf{x}), \mathbf{t}),$$

where $\prec$ denotes the lexicographic ordering over $\mathbb{N} \times \mathbb{N}$, and $\mathbf{tu}$ *denotes the vector obtained by concatenating* $\mathbf{t}$ *and* $\mathbf{u}$.

By the above result, we are now in a position to define, for each l-ary operation f in a compact GSOS system, the effect of applying the function $\mathbf{f}_{\mathsf{ST}}: \mathsf{ST}(\mathsf{Act})^l \to \mathsf{ST}(\mathsf{Act})$ to a vector of trees $\mathbf{t}$ by induction on the relation $\prec$ over the norm given in Definition 6.7. This is done in the following definition.

DEFINITION 6.9. Let $G=(\Sigma_G, R_G)$ be a compact GSOS system, and let f be an l-ary operation in Σ_G. We define the operation $\mathbf{f}_{\mathsf{ST}}: \mathsf{ST}(\mathsf{Act})^l \to \mathsf{ST}(\mathsf{Act})$ inductively by stipulating that, for every $t_1, ..., t_l \in \mathsf{ST}(\mathsf{Act})$:

- $\perp \in \mathbf{f}_{\mathsf{ST}}(t_1, ..., t_l)$ iff $f=\Omega$ or there is an argument i for f such that f tests its ith argument and $\perp \in t_i$;
- $\langle c, t\rangle \in \mathbf{f}_{\mathsf{ST}}(t_1, ..., t_l)$ iff there exist a rule for f of the form (1) and a vector of trees $\mathbf{u}=u_{11}\cdots u_{1m_1}\cdots u_{l1}\cdots u_{lm_l}$ such that:

1. for all $1\leqslant i\leqslant l$, $1\leqslant j\leqslant m_i$, $\langle a_{ij}, u_{ij}\rangle \in t_i$;
2. for all $1\leqslant i\leqslant l$ with $n_i>0$, $\perp \notin t_i$ and, for all $1\leqslant k\leqslant n_i$, $\langle b_{ik}, u\rangle \in t_i$ for no $u\in \mathsf{ST}(\mathsf{Act})$; and
3. $\mathbf{C}_{\mathsf{ST}}[\mathbf{t}, \mathbf{u}]=t$, where $\mathbf{C}_{\mathsf{ST}}$ denotes the derived semantic operation associated with the Σ_G-context $C[\mathbf{x}, \mathbf{y}]$. If $C[\mathbf{x}]$ is a variable x_i, then $\mathbf{C}_{\mathsf{ST}}[\mathbf{t}]=t_i$.

In the above definition, we have inductively defined the synchronization tree corresponding to $\mathbf{f}_{\mathsf{ST}}(\mathbf{t})$ assuming that we have already constructed the compact element $\mathbf{C}_{\mathsf{ST}}[\mathbf{t}, \mathbf{u}]$ needed in clause 3 above. This is justified by Proposition 6.8.

EXAMPLE. When applied to the language preACP$_{\Omega\theta}$, the construction in Definition 6.9 produces the following functions:

- $\delta_{\mathsf{ST}}=\varnothing$,
- $\Omega_{\mathsf{ST}}=\{\perp\}$,
- for every $t\in \mathsf{ST}(\mathsf{Act})$, $\mathbf{a}_{\mathsf{ST}}(t)=\{\langle a, t\rangle\}$,
- for every $t_1, t_2 \in \mathsf{ST}(\mathsf{Act})$, $t_1 +_{\mathsf{ST}} t_2 = t_1 \cup t_2$,
- for every $t_1, t_2 \in \mathsf{ST}(\mathit{Act})$, $t_1 \parallel_{\mathsf{ST}} t_2$ is given by

1. $\perp \in t_1 \parallel_{\mathsf{ST}} t_2$ iff $\perp \in t_1$ or $\perp \in t_2$;
2. $\langle c, t\rangle \in t_1 \parallel_{\mathsf{ST}} t_2$ iff one of the following holds:

(a) there exists $\langle c, t_1'\rangle \in t_1$ such that $t = t_1' \parallel_{\mathsf{ST}} t_2$, or

(b) there exists $\langle c, t_2'\rangle \in t_2$ such that $t_1 = t_1 \parallel_{\mathsf{ST}} t_2'$, or

(c) there exist $\langle a, t_1'\rangle \in t_1$ and $\langle b, t_2'\rangle \in t_2$ such that $c=\gamma(a, b)$ and $t = t_1' \parallel_{\mathsf{ST}} t_2'$.

- for every $t\in \mathsf{ST}(\mathsf{Act})$, $\theta_{\mathsf{ST}}(t)$ is given by

1. $\perp \in \theta_{\mathsf{ST}}(t)$ iff $\perp \in t$,
2. $\langle c, t_1\rangle \in \theta_{\mathsf{ST}}(t)$ iff there exists $\langle c, t'\rangle \in t$ such that:

(a) either c is maximal in $(\mathsf{Act}, >)$ and $\theta_{\mathsf{ST}}(t')=t_1$,

(b) or c is not maximal in $(\mathsf{Act}, >)$, for no action $b>c$ and synchronization tree t_2 $\langle b, t_2\rangle \in t$, $\perp \notin t$, and $\theta_{\mathsf{ST}}(t')=t_1$.

The reader familiar with [48] will notice the similarity of these definitions to those given in that reference. (*End of Example*)

We shall now prove that the definition of the operations given in Definition 6.9 endows the preorder of synchronization trees $\mathsf{ST}(\mathsf{Act})$ with a Σ_G-preorder structure. To this end, it is sufficient to prove that each operation $\mathbf{f}_{\mathsf{ST}}$ is monotonic with respect to the preorder $\sqsubseteq$ defined in Section 5.2.

THEOREM 6.10 (Monotonicity). *Let G be a compact GSOS system, and let f be an l-ary operation in Σ_G. Then the function $\mathbf{f}_{\mathsf{ST}}$ given by the construction in Definition 6.9 is monotonic with respect to $\sqsubseteq$.*

Proof. We prove that for each Σ_G-context $C[\mathbf{x}]$ in which each variable occurs at most once, and for vectors of synchronization trees $\mathbf{t}$ and $\mathbf{u}$ of the appropriate length,

$$\mathbf{t}\sqsubseteq \mathbf{u} \Rightarrow \mathbf{C}_{\mathsf{ST}}[\mathbf{t}] \sqsubseteq \mathbf{C}_{\mathsf{ST}}[\mathbf{u}]. \tag{16}$$

To show the above statement, we associate with each triple $(C[\mathbf{x}], \mathbf{t}, \mathbf{u})$ the pair of natural numbers

$$\Big(W(C[\mathbf{x}]), \sum\{\mathsf{ht}(t_i)+\mathsf{ht}(u_i) \mid x_i \text{ occurs in } C[\mathbf{x}]\}\Big).$$

We then prove that (16) holds by well-founded induction on such pairs ordered lexicographically. The details of the proof are similar to those in the proof of Theorem 3.9 and may be found in [9]. ∎

Because of the above result, the construction of Definition 6.9 allows us to define, for each operation symbol f in the signature of a compact GSOS system, a monotonic function $\mathbf{f}_{\mathsf{ST}}$ over the preorder of finite synchronization trees $\mathsf{ST}(\mathsf{Act})$ of the appropriate arity. This is exactly what is needed to endow the preorder $(\mathsf{ST}(\mathsf{Act}), \sqsubseteq)$ with the structure of a Σ_G-preorder. We now proceed to show that, for any compact GSOS system G, the denotational semantics induced by $\mathscr{K}(\mathscr{D})[\![\cdot]\!]$ (see Section 5.2) for recursion-free terms in $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$ is fully abstract with respect to the bisimulation preorder; i.e., that for all $P, Q\in \mathrm{T}(\Sigma_G)$,

$$P \lesssim Q \Leftrightarrow \mathscr{K}(\mathscr{D})[\![P]\!] \sqsubseteq \mathscr{K}(\mathscr{D})[\![Q]\!].$$

First of all, we relate the operational semantics of recursion-free terms to the transition system view of $\mathsf{ST}(\mathsf{Act})$.

LEMMA 6.11. *Let $G=(\Sigma_G, R_G)$ be a compact GSOS system. Then, for all $P\in T(\Sigma_G)$, the following statements hold*:

1. *$P\downarrow_G$ iff $\mathsf{ST}[\![P]\!]\downarrow$;*
2. *if $P \xrightarrow{a}_G Q$ then $\mathsf{ST}[\![P]\!] \xrightarrow{a} \mathsf{ST}[\![Q]\!]$;*

3. *if* $\mathsf{ST}[\![P]\!] \xrightarrow{a} t$ *then* $P \xrightarrow{a}_G Q$ *for some* Q *such that* $\mathsf{ST}[\![Q]\!] = t$.

Proof. All the statements can be easily seen to hold by structural induction on P. Statements 2 and 3 must be proven simultaneously. The details are standard, and therefore we omit them. ▮

PROPOSITION 6.12. *Let* $G = (\Sigma_G, R_G)$ *be a compact GSOS system. Then, for all* $P \in T(\Sigma_G)$, $P \sim \mathsf{ST}[\![P]\!]$.

Proof. Lemma 6.11 tells us that the symmetric closure of the relation

$$\mathfrak{R} = \{(P, \mathsf{ST}[\![P]\!]) \mid P \in \mathrm{T}(\Sigma_G)\}$$

is a prebisimulation. ▮

The results we have established so far allow us to prove that our denotational semantics is fully abstract with respect to the bisimulation preorder for recursion-free terms.

THEOREM 6.13 (Full Abstraction for Recursion-free Terms). *Let G be a compact GSOS system. Then, for all* $P, Q \in T(\Sigma_G)$, $P \lesssim Q$ *iff* $\mathscr{K}(\mathscr{D})[\![P]\!] \sqsubseteq \mathscr{K}(\mathbb{D})[\![Q]\!]$.

Proof. We reason as follows:

$$\begin{aligned}
P \lesssim Q &\Leftrightarrow \mathsf{ST}[\![P]\!] \lesssim \mathsf{ST}[\![Q]\!] && \text{(Proposition 6.12)}\\
&\Leftrightarrow \mathsf{ST}[\![P]\!] \sqsubseteq \mathsf{ST}[\![Q]\!] && \text{(Fact 5.3)}\\
&\Leftrightarrow (\mathsf{ST}[\![P]\!])^{\mathrm{c}} \sqsubseteq (\mathsf{ST}[\![Q]\!])^{\mathrm{c}} && \text{(Lemma 5.7(2))}\\
&\Leftrightarrow \mathscr{K}(\mathscr{D})[\![P]\!] \sqsubseteq \mathscr{K}(\mathscr{D})[\![Q]\!] && \text{(Corollary 5.9(2)).}
\end{aligned}$$ ▮

Our aim in the remainder of this section will be to extend the above full abstraction result to the whole of $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$, for any compact GSOS system G. First of all, in order to define an interpretation of programs in $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$ as elements of $\mathscr{D}$, we need to define a continuous Σ_G-algebra structure on $\mathscr{D}$. As $(\mathscr{D}, \sqsubseteq_{\mathscr{D}})$ is, up to isomorphism, the unique algebraic cpo with $(\mathscr{K}(\mathscr{D}), \sqsubseteq)$ as poset of compact elements, this is easily done by using Eq. (14) to define a continuous function $\mathbf{f}_{\mathscr{D}}$ for each $f \in \Sigma_G$. By the general theory of algebraic semantics (see, e.g., [42]), we then have that, for all $P, Q \in \mathrm{T}(\Sigma_G)$,

$$\mathscr{D}[\![P]\!] \sqsubseteq_{\mathscr{D}} \mathscr{D}[\![Q]\!] \Leftrightarrow \mathscr{K}(\mathscr{D})[\![P]\!] \sqsubseteq \mathscr{K}(\mathscr{D})[\![Q]\!]. \quad (17)$$

In view of Theorem 6.13, our desired full abstraction result will follow if we prove that the behavioural preorder $\lesssim_\omega$ is algebraic. This is because, from Lemma 5.1 and the fact that, by our constructions, each term $P \in \mathrm{T}(\Sigma_G)$ is interpreted as a compact element of $\mathscr{D}$, the relation $\sqsubseteq_{\mathscr{D}}$ is algebraic, and two algebraic relations that coincide over $\mathrm{T}(\Sigma_G)$ do, in fact, coincide over the whole of $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$. The key to the proof of the algebraicity of $\lesssim_\omega$ is the following general theorem providing a partial completeness result for $\lesssim$ in the sense of Hennessy [8, 39] for arbitrary compact GSOS systems.

Before stating the partial completeness theorem, we introduce a technical notion from [7] which will be useful in the remainder of this paper.

DEFINITION 6.14. A GSOS system H is a *disjoint extension* of a GSOS system G if the signature and rules of H include those of G, and H introduces no new rules for operations of G.

Note that the relation "is a disjoint extension of" is a partial order over the set of GSOS systems. Moreover, if H disjointly extends G then it is not hard to see that:

- for every program P in $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$, $P\downarrow_{Grec}$ iff $P\downarrow_{Hrec}$;
- for all $P, Q \in \mathsf{CREC}(\Sigma_G, \mathsf{PVar})$, $P \xrightarrow{a}_{Grec} Q$ implies $P \xrightarrow{a}_{Hrec} Q$; and
- for all $P \in \mathsf{CREC}(\Sigma_G, \mathsf{PVar})$, $Q \in \mathsf{CREC}(\Sigma_H, \mathsf{PVar})$, $P \xrightarrow{a}_{Hrec} Q$ implies $Q \in \mathsf{CREC}(\Sigma_G, \mathsf{PVar})$ and $P \xrightarrow{a}_{Grec} Q$.

This means in particular that, for $P, Q \in \mathsf{CREC}(\Sigma_G, \mathsf{PVar})$, $P \lesssim_{Grec} Q \Leftrightarrow P \lesssim_{Hrec} Q$.

THEOREM 6.15 (Partial Completeness). *Let G be a compact GSOS system. Then there exist a compact GSOS system H and a set of* Σ_H*-inequations T such that*:

- *H disjointly extends G and* $\mathsf{FINTREE}_\Omega$, *and*
- *for all* $P \in T(\Sigma_H)$, $Q \in \mathsf{CREC}(\Sigma_H, \mathsf{PVar})$, $P \lesssim_{Hrec} Q$ *iff* $T \cup \{(4)\} \vdash P \leqslant Q$.

The proof of this theorem is rather involved and occupies the whole of the next section. It involves giving an algorithm for finding a complete axiomatization of the relation $\lesssim$ over a disjoint extension of our original language following the lines of [7]. As the details of the proof are not necessary to understand the developments of this section, we feel free to assume Theorem 6.15 in what follows and defer its proof. Apart from its intrinsic interest, the main consequence of Theorem 6.15 is the following key result that essentially states that, for any compact GSOS system, finite trees (and therefore recursion-free terms) play the role of compact elements with respect to the preorder $\lesssim$.

THEOREM 6.16. *Let G be a compact GSOS system. Suppose that t is a synchronization tree in* $\mathsf{ST}(\mathsf{Act})$ *and that* $P \in \mathsf{CREC}(\Sigma_G, \mathsf{PVar})$. *Then* $t \lesssim_{Grec} P$ *iff there exists a finite approximation* P^n *of P such that* $t \lesssim_{Grec} P^n$.

Proof. The "if" implication follows immediately from the fact that $P^n \leqslant_\Omega P$ implies $P^n \lesssim_{Grec} P$; so we concentrate on the proof of the "only if" implication. The proof relies on

general properties of initial continuous algebras in inequational varieties that may be found in, e.g., [42].

Let G be a compact GSOS system. Then, by Theorem 6.15, there exist a compact GSOS system H that disjointly extends G and $\mathsf{FINTREE}_\Omega$, and a collection T of Σ_H-inequations such that, for all $P \in \mathrm{T}(\Sigma_H)$, $Q \in \mathsf{CREC}(\Sigma_H, \mathsf{PVar})$, $P \lesssim_{Hrec} Q$ iff $T \cup \{(4)\} \vdash P \leqslant Q$. Let CI_T denote the initial continuous Σ_H-algebra that satisfies the set of Σ_H-inequations T. Then, from the general theory of algebraic semantics (cf., e.g., [42, Theorem 4.3.4]), we have that, for all $P \in \mathrm{T}(\Sigma_H)$, $Q \in \mathsf{CREC}(\Sigma_H, \mathsf{PVar})$,

$$\mathrm{CI}_T[\![P]\!] \leqslant \mathrm{CI}_T[\![Q]\!] \Leftrightarrow T \cup \{(4)\} \vdash P \leqslant Q. \qquad (18)$$

Let $t \in \mathsf{ST}(\mathsf{Act})$ and $P \in \mathsf{CREC}(\Sigma_G, \mathsf{PVar})$. Assume that $t \lesssim_{Grec} P$. As H is a disjoint extension of G, we have that $t \lesssim_{Hrec} P$. Moreover, as H disjointly extends $\mathsf{FINTREE}_\Omega$, there exists a term $Q_t \in \mathrm{T}(\Sigma_H)$ such that $Q_t \sim_{Hrec} t \lesssim_{Hrec} P$. By Theorem 6.15, it follows that $T \cup \{(4)\} \vdash Q_t \leqslant P$. By (18), we then have that $\mathrm{CI}_T[\![Q_t]\!] \leqslant \mathrm{CI}_T[\![P]\!]$. By the construction of CI_T, the denotation of every recursion-free term in $\mathsf{CREC}(\Sigma_H, \mathsf{PVar})$ is a compact element in CI_T. Using (9), this implies that $\mathrm{CI}_T[\![Q_t]\!] \leqslant \mathrm{CI}_T[\![P^n]\!]$ for some finite approximation P^n of P. Applying (18) and the soundness of the inequations in T with respect to $\lesssim_{Hrec}$, we have that $t \sim_{Hrec} Q_t \lesssim_{Hrec} P^n$, As P^n is a Σ_G-term, and H disjointly extends G, we finally conclude that $t \lesssim_{Grec} P^n$, as required. ∎

The above result, in conjunction with Proposition 4.6, allows us to prove that $\lesssim_\omega$ is indeed algebraic.

THEOREM 6.17. *Let G be a compact GSOS system. Then the relation $\lesssim_\omega$ over $\mathsf{lts}(Grec)$ is algebraic.*

Proof. We prove that, for all $P, Q \in \mathsf{CREC}(\Sigma_G, \mathsf{PVar})$, $P \lesssim_\omega Q$ iff $P \lesssim_\omega^A Q$.

- Assume that $P \lesssim_\omega Q$. We prove that $P \lesssim_\omega^A Q$, i.e., that for every finite approximation P^n of P there exists a finite approximation Q^m of Q such that $P^n \lesssim_\omega Q^m$.

Let P^n be a finite approximation of P. Then, as $\lesssim_\omega$ is a Σ_G-precongruence and the defining laws for $\leqslant_\Omega$ are sound with respect to it, we have that $P^n \lesssim_\omega P$. As G is compact and $P^n \in \mathrm{T}(\Sigma_G)$, by Lemma 6.5, there exists a finite synchronization tree t_{P^n} such that $t_{P^n} \sim P^n$. Thus by transitivity of $\lesssim_\omega$, $t_{P^n} \lesssim_\omega Q$. We may now apply Theorem 6.16 to conclude that, for some finite approximation Q^m of Q, $t_{P^n} \lesssim Q^m$. By Lemma 2.5, it follows that $P^n \lesssim_\omega Q^m$, as required.

- Assume that $P \lesssim_\omega^A Q$. We prove that $P \lesssim_\omega Q$. In fact, as $\lesssim_\omega$ coincides with the finitary part of $\lesssim$ by Proposition 4.6, it is sufficient to show that $P \lesssim^{\mathrm{F}} Q$. Assume to this end that $t \lesssim P$ for some $t \in \mathsf{ST}(\mathsf{Act})$. We then reason as follows:

$$\begin{aligned}
t \lesssim P &\Leftrightarrow \exists P^n\colon t \lesssim P^n && \text{(By Theorem 6.16)}\\
&\Leftrightarrow \exists P^n\colon t \lesssim_\omega P^n && \text{(By Lemma 2.5)}\\
&\Rightarrow \exists P^n, Q^m\colon t \lesssim_\omega P^n \lesssim_\omega Q^m && (P \lesssim_\omega^{\mathrm{A}} A)\\
&\Rightarrow t \lesssim_\omega Q && (\leqslant_\Omega \subseteq \lesssim_\omega)\\
&\Leftrightarrow t \lesssim Q && \text{(By Lemma 2.5)}
\end{aligned}$$

Thus $P \lesssim^{\mathrm{F}} Q$, as required. ∎

It is interesting to note that the relation $\lesssim_\omega$ is, in general, not algebraic for arbitrary GSOS systems. Consider, for instance, the GSOS system obtained by adding the constant a^ω given by the rule (15) to our running example $\mathrm{preACP}_{\Omega\theta}$. Because of the presence of a^ω, the resulting GSOS system is *not* compact. We claim that the relation $\lesssim_\omega$ is not algebraic over this language. In fact, consider the terms a^ω and $\mathrm{fix}(X = a.X)$. It is easy to see that $a^\omega \lesssim_\omega \mathrm{fix}(X = a.X)$. Moreover, for every $n \geqslant 1$, $(a^\omega)^n \equiv a^\omega$. However, for no finite approximation $(\mathrm{fix}(X = a.X))^n$ of $\mathrm{fix}(X = a.X)$, it holds that $a^\omega \lesssim_\omega (\mathrm{fix}(X = a.X))^n$. This is because each $(\mathrm{fix}(X = a.X))^n$ has the form $a^n.\Omega$ and $a^\omega \not\lesssim_{n+1} a^n.\Omega$. As highlighted by this example, the non-algebraicity of the preorder $\lesssim_\omega$ for arbitrary GSOS languages is due to the fact that, in general, recursion-free terms need not be "semantically finite."

In light of the above results, we can now show that, for any compact GSOS system G, the denotational semantics for $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$ is fully abstract with respect to $\lesssim_\omega$.

THEOREM 6.18 (Full Abstraction). *Let G be a compact GSOS system. Then, for every $P, Q \in \mathsf{CREC}(\Sigma_G, \mathsf{PVar})$, $P \lesssim_\omega Q$ iff $\mathscr{D}[\![P]\!] \sqsubseteq_{\mathscr{D}} \mathscr{D}[\![Q]\!]$.*

Proof. Let G be a compact GSOS system and $P, Q \in \mathsf{CREC}(\Sigma_G, \mathsf{PVar})$. We proceed as follows:

$$\begin{aligned}
P \lesssim_\omega Q &\Leftrightarrow \forall n \exists m\colon P^n \lesssim_\omega Q^m\\
&\qquad \text{(By Theorem 6.17)}\\
&\Leftrightarrow \forall n \exists m\colon P^n \lesssim Q^m\\
&\qquad \text{(By Corollary 6.5 and Lemma 2.5)}\\
&\Leftrightarrow \forall n \exists m\colon \mathscr{D}[\![P^n]\!] \sqsubseteq_{\mathscr{D}} \mathscr{D}[\![Q^m]\!]\\
&\qquad \text{(By Theorem 6.13)}\\
&\Leftrightarrow \mathscr{D}[\![P]\!] \sqsubseteq \mathscr{D}[\![Q]\!]\\
&\qquad \text{(By property (9)).}
\end{aligned}$$

∎

When applied to the version of SCCS considered by Abramsky in [1], the techniques we have presented deliver a denotational semantics that is in complete agreement with the one given by Abramsky in that paper. This is an easy consequence of Theorem 6.18, Abramsky's full abstraction theorem [1, Theorem 6.19] and the fact that, as remarked

in Section 4, the behavioural semantics for SCCS generated by our methods is exactly the standard semantics given by Hennessy in [39].

7. PARTIAL COMPLETENESS FOR COMPACT GSOS LANGUAGES

Our aim in this section is to give a proof of Theorem 6.15. The main ideas underlying the proof of this partial completeness result are a generalization of those used, e.g., by Hennessy in [39] to establish a similar result for Milner's SCCS [63]. This generalization of Hennessy's approach is achieved along the lines of the developments in [7, 5], but the details are quite different from the ones in the aforementioned references.

Let G be a compact GSOS system. We present an algorithm that can be used to generate a compact GSOS system H that disjointly extends G and $\mathsf{FINTREE}_\Omega$, and a set of Σ_H-inequations such that, for all $P \in \mathrm{T}(\Sigma_H)$ and $Q \in \mathsf{CREC}(\Sigma_H, \mathsf{PVar})$,

$$P \lesssim_{Hrec} Q \Leftrightarrow T \cup \{(\mathrm{Rec})\} \vdash P \leqslant Q \tag{19}$$

where (Rec) stands for Eq. (4), stating that a recursive term is equivalent to its unwinding. As H is a disjoint extension of the GSOS system G we started from, we have that $\lesssim_{Grec}$ coincides with $\lesssim_{Hrec}$ over $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$. A solution to (19) thus solves, in particular, the partial completeness problem for the original language G.

The equational theory T generated by the methods presented in this section will include the following set of inequalities,[6] which will be henceforth referred to as $T_{\mathsf{FINTREE}_\Omega}$:

$$x + y = y + x \tag{20}$$

$$(x + y) + z = x + (y + z) \tag{21}$$

$$x + x = x \tag{22}$$

$$x + \delta = x \tag{23}$$

$$\Omega \leqslant x. \tag{24}$$

It is not difficult to see that the above inequalities are sound with respect to $\lesssim$ in any GSOS system that disjointly extends $\mathsf{FINTREE}_\Omega$. Moreover, it is well-known that they are complete with respect to $\lesssim$ over $\mathsf{FINTREE}_\Omega$. (See, e.g., the results in [39, 8].)

LEMMA 7.1. *Let G be a GSOS system that disjointly extends $\mathsf{FINTREE}_\Omega$. Then the inequational theory $T_{\mathsf{FINTREE}_\Omega}$ is sound with respect to $\lesssim_{Grec}$.*

[6] As usual, an equation $P = Q$ should be read as a shorthand for the pair of inequations $P \leqslant Q$ and $Q \leqslant P$.

In order to prove Theorem 6.15. we aim at generating a compact GSOS system H that disjointly extends G and $\mathsf{FINTREE}_\Omega$, and an inequational theory T over Σ_H that includes $T_{\mathsf{FINTREE}_\Omega}$ and has the following properties:

- *T is Ω-head normalizing for terms in $T(\Sigma_H)$.* Adapting the terminology in [42], an Ω-head normal form is a term of the form $\sum_{i \in I} a_i . P_i[+\Omega]$, where the notation $[+\Omega]$ means that Ω is an optional summand. The inequational theory generated by our methods will have the property that every recursion-free term in $\mathsf{CREC}(\Sigma_H, \mathsf{PVar})$ is provably equal to an Ω-head normal form, i.e.,

$$\forall P \in \mathrm{T}(\Sigma_H)\ \exists \sum_{i \in I} a_i . P_i[+\Omega] : T \vdash P = \sum_{i \in I} a_i . P_i[+\Omega]. \tag{25}$$

- *T is head normalizing for convergent terms in $\mathsf{CREC}(\Sigma_H, \mathsf{PVar})$.* A head normal form is a term of the form $\sum_{i \in I} a_i . P_i$. The inequational theory generated by our methods will have the property that every convergent term in $\mathsf{CREC}(\Sigma_H, \mathsf{PVar})$ is provably equal to a head normal form, i.e., for every $P \in \mathsf{CREC}(\Sigma_H, \mathsf{PVar})$,

$$P{\downarrow_{Hrec}} \Rightarrow \exists \sum_{i \in I} a_i . P_i : T \cup \{(\mathrm{Rec})\} \vdash P = \sum_{i \in I} a_i . P_i. \tag{26}$$

- *T absorbs transitions.* The inequational theory generated by our methods will be such that for all $P \in \mathsf{CREC}(\Sigma_H, \mathsf{PVar})$ there exists a program $P^* \in \mathsf{CREC}(\Sigma_H, \mathsf{PVar})$ with the following properties:

$$T \vdash P = P^* \tag{27}$$

and, for all $a \in \mathsf{Act}$, $Q \in \mathsf{CREC}(\Sigma_H, \mathsf{PVar})$,

$$P^* \xrightarrow{a}_{Hrec} Q \Rightarrow T \cup \{(\mathrm{Rec})\} \vdash P^* = P^* + a . Q. \tag{28}$$

Our interest in inequational theories with the aforementioned properties stems from the following result, from which, after having presented the promised algorithm, we shall be able to derive Theorem 6.15.

THEOREM 7.2. *Let H be a compact GSOS system that disjointly extends $\mathsf{FINTREE}_\Omega$. Suppose that T is a collection of sound inequations with respect to $\lesssim_{Hrec}$ that extends $T_{\mathsf{FINTREE}_\Omega}$. Suppose further that T is Ω-head normalizing for terms in $T(\Sigma_H)$, head normalizing for convergent terms in $\mathsf{CREC}(\Sigma_H, \mathsf{PVar})$ and that T absorbs transitions. Then* (19) *holds for such H and T.*

Proof. Assume that H is a compact GSOS system that disjointly extends $\mathsf{FINTREE}_\Omega$, and that T is a collection of inequations over Σ_H that includes $T_{\mathsf{FINTREE}_\Omega}$, is sound with respect to $\lesssim_{Hrec}$ and has properties (25)–(28). We prove that (19) holds for such an H and T.

The soundness part of the statement is guaranteed to hold by the proviso of the theorem, by Fact 4.5 and by Corollary 4.9. Therefore, we focus on the proof of partial completeness. To this end, note, first of all, that, by Lemma 3.8 and (7), $\mathsf{Der}(P)$ is finite for every $P \in \mathrm{T}(\Sigma_H)$. Moreover, as H is compact, by Proposition 6.4 no term $P \in \mathrm{T}(\Sigma_H)$ can have infinite derivations. Thus we can associate with each $P \in \mathrm{T}(\Sigma_H)$ a natural number $depth(P)$, denoting the maximum number of consecutive transitions possible from P. Note further that, for any two terms $P_1, P_2 \in \mathrm{T}(\Sigma_H)$ that are related by $\sim_{Hrec}$, $depth(P_1) = depth(P_2)$.

Assume now that $P \in \mathrm{T}(\Sigma_H)$ and that $Q \in \mathsf{CREC}(\Sigma_H, \mathsf{PVar})$. Suppose further that $P \lesssim_{Hrec} Q$. We prove, using properties (25)–(28) above, that

$$T \cup \{(\mathrm{Rec})\} \vdash P \leqslant Q. \tag{29}$$

The proof is by induction on $depth(P)$. We assume, as inductive hypothesis, that the claim holds for all $P' \in \mathrm{T}(\Sigma_H)$, $Q' \in \mathsf{CREC}(\Sigma_H, \mathsf{PVar})$ with $P' \lesssim_{Hrec} Q'$ and $depth(P') < depth(P)$, and show that it holds for P and Q.

To show (29), by transitivity, it is sufficient to establish the following two claims:

1. $T \cup \{(\mathrm{Rec})\} \vdash P \leqslant P + Q$, and
2. $T \cup \{(\mathrm{Rec})\} \vdash P + Q \leqslant Q$.

We now proceed by proving the above claims separately. First of all, note that, as $P \in \mathrm{T}(\Sigma_H)$, by (25) we have that P is provably equal to an Ω-head normal form $\sum_{i \in I} a_i . P_i[+\Omega]$, i.e.,

$$t \vdash P = \sum_{i \in I} a_i . P_i[+\Omega]. \tag{30}$$

Proof of Claim 1. We prove that $T \cup \{(\mathrm{Rec})\} \vdash P \leqslant P + Q$ by distinguishing two cases, depending on whether the Ω-head normal form for P has an Ω summand or not.

- Assume that the Ω-head normal form for P has an Ω summand. Then we simply reason as follows:

$$\begin{aligned} T \vdash P &= \sum_{i \in I} a_i . P_i + \Omega \\ &\leqslant \sum_{i \in I} a_i . P_i + \Omega + Q \quad \text{(by (22) and (24))} \\ &= P + Q. \end{aligned}$$

- Assume that the Ω-head normal form for P does not have an Ω summand. Then, by the soundness of T, we have that

$$P \sim_{Hrec} \sum_{i \in I} a_i . P_i.$$

It follows that $P{\downarrow_{Hrec}}$. As $P \lesssim_{Hrec} Q$, it must also be the case that $Q{\downarrow_{Hrec}}$. By (26), Q is provably equal to a head normal form $\sum_{j \in J} b_j . Q_j$ i.e.,

$$T \cup \{(\mathrm{Rec})\} \vdash Q = \sum_{j \in J} b_j . Q_j.$$

As $P \lesssim_{Hrec} Q$, $P{\downarrow_{Hrec}}$, and $Q{\downarrow_{Hrec}}$, for every $j \in J$ there exists $i_j \in I$ such that $a_{ij} = b_j$ and $P_{i_j} \lesssim_{Hrec} Q_j$. Now note that for every $i \in I$, $depth(P_i) < depth(\sum_{i \in I} a_i . P_i) = depth(P)$. Thus the inductive hypothesis can be applied to infer that

$$T \cup \{(\mathrm{Rec})\} \vdash P_{i_j} \leqslant Q_j.$$

Therefore,

$$\begin{aligned} T \cup \{(\mathrm{Rec})\} \vdash P &= \sum_{i \in I} a_i . P_i \\ &= \sum_{i \in I} a_i . P_i + \sum_{j \in J} a_{i_j} . P_{i_j} \\ &\qquad \text{(by repeated use of (22))} \\ &\leqslant \sum_{i \in I} a_i . P_i + \sum_{j \in J} b_j . Q_j \\ &\qquad \text{(by the inductive hypothesis)} \\ &= P + Q. \end{aligned}$$

This completes the proof of the first claim.

Proof of Claim 2. We now complete the proof by showing that $T \cup \{(\mathrm{Rec})\} \vdash P + Q \leqslant Q$. First of all, note that, as T absorbs transitions, there exists a program $Q^* \in \mathsf{CREC}(\Sigma_H, \mathsf{PVar})$ such that

$$T \vdash Q = Q^* \tag{31}$$

and, for all $a \in \mathsf{Act}$ and $Q' \in \mathsf{CREC}(\Sigma_H, \mathsf{PVar})$,

$$Q^* \xrightarrow{a}_{Hrec} Q' \Rightarrow T \cup \{(\mathrm{Rec})\} \vdash Q^* = Q^* + q.Q'. \tag{32}$$

By the soundness of the inequations in T and (30), we have that

$$\sum_{i \in I} a_i . P_i[+\Omega] \sim_{Hrec} P \lesssim_{Hrec} Q \sim_{Hrec} Q^*.$$

Thus, for every $i \in I$ there exists a term Q_i such that $Q^* \xrightarrow{a_i}_{Hrec} Q_i$ and $P_i \lesssim_{Hrec} Q_i$. By induction, we infer that, for each such pair of processes (P_i, Q_i),

$$T \cup \{(\mathrm{Rec})\} \vdash P_i \leqslant Q_i. \tag{33}$$

Therefore,

$$\begin{aligned} T \cup \{(Rec)\} \vdash P + Q &= P + Q^* \quad \text{(by (31))} \\ &= \sum_{i \in I} a_i . P_i[+\Omega] + Q^* \\ &\leqslant \sum_{i \in I} a_i . P_i[+Q^*] + Q^* \\ &\qquad \text{(by possibly using (24))} \\ &\leqslant \sum_{i \in I} a_i . Q_i + Q^* \\ &\qquad \text{(by (33) and possibly} \\ &\qquad \text{using (22))} \\ &= Q^* \quad \text{(by repeated use of (32))} \\ &= Q \quad \text{(again by (31)).} \end{aligned}$$

The proof of the theorem is now complete. ▮

By the above theorem, all that is needed to show Theorem 6.15 is an algorithm that, starting from a compact GSOS system G, allows us to generate a suitable disjoint extension H of G, and an inequational theory T that enjoys properties (25)–(28). This we now present, following the developments in [5, 7] closely. We take the liberty of referring the reader to those references for detailed motivations for some of the technical definitions to follow.

We now present the core of our strategy for axiomatizing GSOS operations. Following [7], we proceed in two steps: first, we show how to axiomatize a class of particularly well-behaved operations, the smooth operations introduced in Section 7.1. Second, we extend our results to arbitrary GSOS operations in Section 7.2.

Notation 7.3. For any term P in a GSOS system G that disjointly extends $\mathsf{FINTREE}_\Omega$, the notation $P[+\Omega \Leftrightarrow \mathit{Condition}]$ will stand for $P + \Omega$ if Condition is true, and P otherwise.

7.1. *Smooth Operations*

In this subsection we give a way of generating an inequational theory that enjoys properties (25)–(28) for the class of *smooth* operations, using the still simpler *weakly distinctive* operations [23, 5] as a base case. The following definition is from [7], where motivation and examples of smooth operations may be found.

Definition 7.4. A GSOS rule is smooth if it takes the form

$$\frac{\{x_i \xrightarrow{a_i} y_i \mid i \in I\} \cup \{x_i \not\xrightarrow{b_{ij}} \mid i \in K, 1 \leqslant j \leqslant n_i\}}{f(x_1, ..., x_l) \xrightarrow{c} C[\mathbf{x}, \mathbf{y}]} \quad (34)$$

where I, K are disjoint sets such that $I \cup K = \{1, ..., l\}$, and no x_i with $i \in I$ appears in $C[\mathbf{x}, \mathbf{y}]$. An operation from a GSOS system G is smooth if all the rules for this operation are smooth.

For example, the operations in the language $\mathrm{preACP}_{\Omega\theta}$ are smooth, with the possible exception of the priority operation θ, which is only smooth if the priority structure on actions is trivial.

In order to obtain an inequational theory for smooth operations with the required properties, we first show how to obtain equations that describe the interplay between such operations and the $\mathsf{FINTREE}_\Omega$ combinators. Lemmas 7.5–7.11 hold for arbitrary GSOS systems, and will be stated in full generality even though, in the remainder of the paper, we shall only apply them to obtain equations for smooth operations in compact GSOS systems.

Lemma 7.5 (Distributivity Laws). *Let f be an l-ary smooth operation of a GSOS system G that disjointly extends $\mathsf{FINTREE}_\Omega$, and suppose i is an argument of f for which* each *rule for f has a positive antecedent. Then f distributes over $+$ in its ith argument; i.e., for every GSOS system H that disjointly extends G,*

$$\begin{aligned} f(X_1, ..., X_i + Y_i, ..., X_l) \sim_{Hrec} & f(X_1, ..., X_i, ..., X_l) \\ & + f(X_1, ..., Y_i, ..., X_l). \quad (35) \end{aligned}$$

Proof. A minor adaptation of the proof of [4, Lemma 5.2]. ▮

When applied to the communication-merge operation from ACP [18] given by the rules (one such rule for each triple of actions (a, b, c) with $\gamma(a, b) \simeq c$);

$$\frac{x \xrightarrow{a} x',\ y \xrightarrow{b} y'}{x \mid y \xrightarrow{c} x' \mid y'}$$

the above lemma gives the equations

$$\begin{aligned} (X + Y) \mid Z &= X \mid Z + Y \mid Z \\ X \mid (Y + Z) &= X \mid Y + X \mid Z. \end{aligned}$$

We now present lemmas that give *divergence laws* and *inaction laws* to describe the interaction between arbitrary operations and the $\mathsf{FINTREE}_\Omega$ constants Ω and δ, respectively; that is, laws which say when a term $f(\mathbf{P})$ is equivalent to Ω or δ. The following lemmas can certainly be generalized to yield more laws, but they will be enough for our purposes in this study.

Lemma 7.6 (Divergence Laws). *Suppose f is an l-ary smooth operation of a GSOS system G that disjointly extends $\mathsf{FINTREE}_\Omega$, and suppose that, for $1 \leqslant i \leqslant l$, term P_i is of the*

form δ, Ω, X_i, *or* $X_i + \Omega$. *Suppose further that the following conditions are met*:

1. *for each rule for* f *of the form* (34) *there is an index* i *such that*
   - *either* $i \in I$, *and* $P_i \equiv \delta$ *or* $P_i \equiv \Omega$,
   - *or* $i \in K$, $n_i > 0$ *and* $P_i \equiv \Omega$ *or* $P_i \equiv X_i + \Omega$;

and

2. *for some argument* i *tested by* f, $P_i \equiv \Omega$ *or* $P_i \equiv X_i + \Omega$.

Then for every GSOS system H *that disjointly extends* G,

$$f(\mathbf{P}) \sim_{Hrec} \Omega. \tag{36}$$

Proof. Assume that H is a disjoint extension of G, and consider a substitution $\boldsymbol{\sigma}\colon \mathsf{PVar} \to \mathsf{CREC}(\Sigma_H, \mathsf{PVar})$. Then condition 1 of the lemma ensures that $f(\mathbf{P})\,\boldsymbol{\sigma}$ does not have any transition, and condition 2 ensures that $f(\mathbf{P})\,\boldsymbol{\sigma}\uparrow_{Hrec}$. Therefore $f(\mathbf{P})\,\boldsymbol{\sigma} \sim_{Hrec} \Omega$ as required. ▮

To give the reader an idea of the laws that are generated by the above lemma, we consider the binary smooth option $\triangle$ introduced in [7] to axiomatize the priority operation θ. The definition of this operation, as that of θ, assumes the presence of a partially ordered set of actions $(\mathsf{Act}, >)$. The operation $\triangle$ has rules (one for each $a \in \mathsf{Act}$):

$$\frac{x \xrightarrow{a} x', \quad y \not\xrightarrow{b} \quad (\text{for all } b > a)}{x \triangle y \xrightarrow{a} \theta(x')} \tag{37}$$

We apply the above lemma to generate divergence laws for $\triangle$ by considering the possible forms that its arguments can take.

- *Divergence laws when the first argument is* δ. In this case, the first condition of the statement of Lemma 7.6 is met regardless of the form of the second argument. However, condition 2 must also be met. This is not possible if the partial order on actions is flat, i.e., if no two actions are related by $>$. In that case, no divergence laws are generated by the lemma when the first argument of $\triangle$ is the inactive, convergent term δ. Otherwise, Lemma 7.6 gives the following laws:

$$\delta \triangle \Omega = \Omega$$

$$\delta \triangle (Y + \Omega) = \Omega$$

Note that, in the presence of law (22), the first law is redundant as it is provably equal to a substitution instance of the second.

- *Divergence laws when the first argument is* Ω. In this case, the second argument can be arbitrary, and every law generated by Lemma 7.6 can be obtained as a substitution instance of the law

$$\Omega \triangle Y = \Omega.$$

- *Divergence laws when the first argument is a process variable* X. In this case, Lemma 7.6 produces no divergence law. In fact, as Act is finite, there is at least one action a which is maximal with respect to $>$. The instance of rule (37) for a has no negative premise and condition 1(2) in the statement of the lemma cannot be met for it, no matter what the form of the second argument of $\triangle$ is.

We remark here that the requirement $n_i > 0$ in condition 1(2) of the statement of the lemma is vital for the soundness of the generated equations. Without it, Lemma 7.6 could be used to derive the unsound equations

$$X \triangle \Omega = \Omega.$$

This equation is unsound because, if a is an action in Act which is maximal with respect to $>$, then

$$a.\delta \triangle \Omega \sim a.\delta[\,+\Omega \Leftrightarrow (\mathsf{Act}, >) \text{ is not flat}\,] \not\sim \Omega.$$

- *Divergence laws when the first argument is of the form* $X + \Omega$. Reasoning as in the previous case, it is not hard to argue that Lemma 7.6 produces no divergence law.

LEMMA 7.7 (Inaction Laws). *Suppose* f *is an* l*-ary smooth operation of a GSOS system* G *that disjointly extends* $\mathsf{FINTREE}_\Omega$, *and suppose that, for* $1 \leqslant i \leqslant l$, *term* P_i *is of the form* δ, X_i *or* $\sum_{n \in N} c_n.X_n$. *Suppose further that the following conditions are met*:

1. *for each rule for* f *of the form* (34) *there is an index* i *such that either* (1) $i \in I$ *and* $P_i \equiv \delta$ *or* $P_i \equiv \sum_{n \in N} c_n.X_n$ *and* $a_i \notin \{c_n \mid n \in N\}$, *or* (2) $i \in K$, $P_i \equiv \sum_{n \in N} c_n.X_n$ *and there exist* $n \in N$ *and* $1 \leqslant j \leqslant n_i$ *with* $c_n = b_{ij}$; *and*

2. *for no argument* i *tested by* f, *is* P_i *a process variable.*

Then, for every GSOS system H *that disjointly extends* G,

$$f(\mathbf{P}) \sim_{Hrec} \delta. \tag{38}$$

Proof. Assume that H is a disjoint extension of G, and consider a substitution $\boldsymbol{\sigma}\colon \mathsf{PVar} \to \mathsf{CREC}(\Sigma_H, \mathsf{PVar})$. Then condition 1 of the lemma ensures that $f(\mathbf{P})\,\boldsymbol{\sigma}$ does not have any transition, and condition 2 ensures that $f(\mathbf{P})\,\boldsymbol{\sigma}\downarrow_{Hrec}$. Therefore $f(\mathbf{P})\,\boldsymbol{\sigma} \sim_{Hrec} \delta$, as required. ▮

Again, we show the above lemma in action by applying it to generate inaction laws for the $\triangle$ operation. As before, we proceed by considering the possible forms the arguments of $\triangle$ may take.

• *Inaction laws when the first argument of* $\triangle$ *is* δ. In this case, condition 1 in the statement of the lemma is always satisfied. If the partial ordering on Act is flat, then $\triangle$ does not test its second argument, which may take any of the forms specified in Lemma 7.7; all the equations generated by the lemma in this case may be obtained as substitution instances of the law

$$\delta \triangle Y = \delta.$$

Otherwise, $\triangle$ does test its second argument, and, by condition 2, the second argument must be of the form $\sum_{n \in N} a_n . X_n$. (Recall that $\delta \equiv \sum_{n \in \varnothing} a_n . X_n$.) The corresponding law is

$$\delta \triangle \sum_{n \in N} a_n . X_n = \delta.$$

• *Inaction laws when the first argument of* $\triangle$ *is a process variable* X. This case is ruled out by condition 2 of the lemma, and no equations are generated.

• *Inaction laws when the first argument of* $\triangle$ *is of the form* $\sum_{n \in N} a_n . X_n$ *for non-empty* N. In this case, in order to meet condition 1 in the statement of the lemma, the second argument must be of the form $\sum_{m \in M} b_m . Y_m$ and for every $n \in N$ there exists $m \in M$ with $b_m > a_n$. For every pair of such terms, Lemma 38 generates the equation

$$\sum_{n \in N} a_n . X_n \triangle \sum_{m \in M} b_m . Y_m = \delta.$$

We now derive *action laws*, which tell when a process can take an action. These laws will be given for a sub-class of smooth GSOS operations that test their arguments positively in a consistent way. This class of smooth operations is characterized in the following definition.

DEFINITION 7.8. Let f be an l-ary smooth operation in a GSOS system G. We say that f is *weakly distinctive* iff every rule for f tests the same arguments positively.

For a weakly distinctive, smooth operation f in a GSOS system G, we write $\mathsf{Test}^+(f)$ for the set of arguments tested positively by every rule for f, and $\mathsf{Test}^-(f)$ for the set of arguments tested negatively by some rule for f.

For example, the operation $a._-$ for $\mathrm{preACP}_{\Omega\theta}$ and the left-merge and communication-merge operations from ACP [18] are weakly distinctive. As remarked previously, the priority operation θ is smooth iff the poset of actions $(\mathsf{Act}, >)$ is flat. In that case, θ is also weakly distinctive, and uninteresting. Note also that, for a smooth operation f, $\mathsf{Test}^+(f)$ and $\mathsf{Test}^-(f)$ are disjoint.

The notion of weak distinctiveness given above is a weakening of the notion of distinctiveness introduced in [7]. Its definition differs slightly from that given in [4] in that we do not require that the operation f be positive, i.e., that rules for f not have negative antecedents, and consistent in the sense of [4, Definition 7.1].

Notation 7.9. We write $\mathsf{Act}_\perp$ for the set $\mathsf{Act} \cup \{\perp\}$, where $\perp$ is a symbol not occurring in Act.

For a term P of the form $\sum_{i \in I} a_i . P_i [+\Omega]$, $\mathit{Initials}(P)$ will denote the subset of $\mathsf{Act}_\perp$ given by $\{a_i \mid i \in I\} [\cup \{\perp\}]$, where $\perp \in \mathit{Initials}(P)$ iff Ω is a summand of P.

DEFINITION 7.10. Let f be an l-ary weakly distinctive, smooth operation in a GSOS system G. We say that a vector $\langle e_1, ..., e_l \rangle$ over $\mathsf{Act} \cup 2^{\mathsf{Act}_\perp}$ is *consistent* with f iff for every argument i of f, if $i \in \mathsf{Test}^+(f)$ then $e_i \in \mathsf{Act}$, and $e_i \subseteq \mathsf{Act}_\perp$ otherwise.

Let f be an l-ary weakly distinctive, smooth operation in a GSOS system G, and let the vector $\langle e_1, ..., e_l \rangle$ be consistent with f. Then $R_G(f, \langle e_1, ..., e_l \rangle)$ will denote the set of rules $r \in R_G$ of the form (34) with f as principal operation such that

$$r \in R_G(f, \langle e_1, ..., e_l \rangle) \Leftrightarrow$$
$$(1) \quad \forall i \in \mathsf{Test}^+(f) . a_i = e_i, \text{ and}$$
$$(2) \quad \forall i \in K . n_i > 0 \Rightarrow (e_i \cap \{b_{ij} \mid 1 \leqslant j \leqslant n_i\} = \varnothing \text{ and } \perp \notin e_i).$$

Intuitively, $R_G(f, \langle e_1, ..., e_l \rangle)$ is the set of rules for f that can be possibly used to derive outgoing transitions from a term of the form $f(\mathbf{P})$ where, for each argument i for f:

1. if $i \in \mathsf{Test}^+(f)$, then P_i can initially perform an e_i-action, and

2. if $i \in \mathsf{Test}^-(f)$, then P_i converges (encoded by the absence of $\perp$ in the set e_i) and the set of initial actions that P_i can perform is exactly e_i.

For example, any tuple of the form $\langle a, B \rangle$, where $a \in \mathsf{Act}$ and $B \subseteq \mathsf{Act}_\perp$ is consistent with the $\triangle$ operation. For any GSOS system G containing this operation, the set $R_G(\triangle, \langle a, B \rangle)$ is a singleton if a is maximal in the poset $(\mathsf{Act}, >)$ or $\perp \notin B$ and B does not contain any $b > a$, and it is empty otherwise. In fact, for any operation f that, like $\triangle$, is smooth and distinctive in the sense of [7], the set $R_G(f, \langle e_1, ..., e_l \rangle)$ is either empty or it contains a single rule. This property does not hold in general for weakly distinctive operations. As an example, consider a GSOS system G that contains the smooth, distinctive operation $\oplus$ [26, 32, 40, 42] given by the rules

$$\frac{}{x \oplus y \xrightarrow{\tau} x} \qquad \frac{}{x \oplus y \xrightarrow{\tau} y}$$

Then the vector $\langle \varnothing, \varnothing \rangle$ is consistent with $\oplus$, and $R_G(\oplus, \langle \varnothing, \varnothing \rangle)$ contains both the above rules.

LEMMA 7.11. *Suppose that f is a weakly distinctive smooth operation of arity l of a disjoint extension G of* $\mathsf{FINTREE}_\Omega$. *Let $\langle e_1, ..., e_l\rangle$ be a vector over $\mathsf{Act} \cup 2^{\mathsf{Act}_\perp}$ that is consistent with f. Finally, let* $\mathbf{P}$ *be the vector of terms given by*

$$P_i \equiv \begin{cases} e_i.Y_i & i \in \mathsf{Test}^+(f) \\ \sum \{b.Z_b \mid b \in e_i\}[+\Omega \Leftrightarrow \perp \in e_i] & i \in \mathsf{Test}^-(f) \\ X_i & \text{otherwise,} \end{cases}$$

where all the process variables are distinct. Then, for every disjoint extension H of G,

$$f(\mathbf{P}) \sim_{Hrec} \sum_{r \in R_G(f, \langle e_1, ..., e_l\rangle)} \mathsf{action}(r).\mathsf{target}(r)\{\mathbf{P}/\mathbf{x}, \mathbf{Y}/\mathbf{y}\} \times [+\Omega \Leftrightarrow (\exists i \in \mathsf{Test}^-(f): \perp \in e_i) \vee f = \Omega]. \quad (39)$$

Proof. Assume that H is a disjoint extension of G, and consider a substitution $\boldsymbol{\sigma}: \mathsf{PVar} \to \mathsf{CREC}(\Sigma_H, \mathsf{PVar})$. By the definition of $\mathbf{P}$ and the fact that f is smooth, it follows immediately that a transition of the form $f(\mathbf{P})\,\boldsymbol{\sigma} \xrightarrow{c}_{Hrec} Q$ holds iff there exists a rule r in $R_G(f, \langle e_1, ..., e_l\rangle)$ with action c such that

$$\mathsf{target}(r)\{\mathbf{P}/\mathbf{x}, \mathbf{Y}/\mathbf{y}\}\,\boldsymbol{\sigma} = Q.$$

Similarly, the form of Eq. (39) ensures that $f(\mathbf{P})\,\boldsymbol{\sigma}$ converges iff the instantiated right-hand side of the equation does. ∎

As an example, we apply the above lemma to derive action laws for the $\triangle$ operation specified by (37). As remarked previously, any vector over $\mathsf{Act} \cup 2^{\mathsf{Act}_\perp}$ that is consistent with $\triangle$ has the form $\langle a, B\rangle$ for some $a \in \mathsf{Act}$ and $B \subseteq \mathsf{Act}_\perp$. We proceed to generate the corresponding action law for $\triangle$ by distinguishing two cases, depending on whether a is maximal in $(\mathsf{Act}, >)$ or not.

- If a is maximal in $(\mathsf{Act}, >)$, then $R_G(\triangle, \langle a, B\rangle)$ is the singleton set containing the only rule for $\triangle$ with action a. The equation generated by Lemma 7.11 then takes the form

$$a.Y \triangle \sum_{b \in B} b.Z_b[+\Omega] = a.\theta(Y)[+\Omega],$$

where Ω is a summand of the right-hand side of the equation iff it appears as a summand of the second argument iff $\perp \in B$.

- If a is *not* maximal in $(\mathsf{Act}, >)$, then the form of the equation produced by Lemma 7.11 depends on whether there is a reason in the set B that prevents the application of the only rule for $\triangle$ to derive a transition from the term $a.Y \triangle \sum_{b \in B} b.Z_b[+\Omega]$, or not

  — If $b > a$ for some $b \in B$ or $\perp \in B$, then $R_G(\triangle, \langle a, B\rangle)$ is empty. The equation given by Lemma 7.11 in this case is then

$$a.Y \triangle \sum_{b \in B} b.Z_b[+\Omega] = \delta[+\Omega],$$

where Ω is a summand of the right-hand side of the equation iff it appears as a summand of the second argument iff $\perp \in B$.[7]

  — Otherwise, $R_G(\triangle, \langle a, B\rangle)$ is the singleton set containing the only rule for $\triangle$ with action a, and Lemma 7.11 gives the equation

$$a.Y \triangle \sum_{b \in B} b.Z_b = a.\theta(Y).$$

The equations given by the above lemmas provide us with an equational theory that is strong enough to establish properties (25)–(28) for terms built from the $\mathsf{FINTREE}_\Omega$ operations and weakly distinctive smooth operations. For the sake of clarity, we reiterate below the definitions of the types of head normal forms that we shall consider in the remainder of the paper.

DEFINITION 7.12. A term P over a signature $\Sigma \supseteq \Sigma_{\mathsf{FINTREE}_\Omega}$ is in *Ω-head normal form* if it is of the form $\sum a_i.P_i[+\Omega]$. We say that P is in *head normal form* if it is of the form $\sum a_i.P_i$.

We prove, first of all, that the equations generated by Lemmas 7.5–7.11 allow us to associate a Ω-head normal form with each recursion-free program built from the $\mathsf{FINTREE}_\Omega$ operations and weakly distinctive smooth operations.

THEOREM 7.13. *Suppose G is a GSOS system that disjointly extends* $\mathsf{FINTREE}_\Omega$. *Let $\Sigma \subseteq \Sigma_G - \Sigma_{\mathsf{FINTREE}_\Omega}$ be a collection of weakly distinctive smooth operations of G. Let T be the equational theory that extends $T_{\mathsf{FINTREE}_\Omega} \setminus \{(24)\}$ with the following axioms, for each operation f from Σ:*

1. *for each argument i of f that is tested positively by f, a distributivity axiom* (35),
2. *for each vector $\langle e_1, ..., e_l\rangle$ over $\mathsf{Act} \cup 2^{\mathsf{Act}_\perp}$ consistent with f an action law* (39),
3. *all the divergence laws* (36) *for f, and*
4. *all the inaction laws* (38) *for f.*

[7] As shown by this example, Lemma 7.11 also delivers inaction and divergence laws. For this reason, the name "action lemma" we have used for it is a bit misleading. We hope that this does not cause the reader any confusion.

Then the following statements hold:

- *If* $T \vdash P = Q$, *then, for every disjoint extension* H *of* G, $P \sim_{Hrec} Q$.
- *For every* $P \in T(\Sigma \cup \Sigma_{\mathsf{FINTREE}_\Omega})$, *there exists an* Ω*-head normal form* Ω*-hnf*(P) *such that*

$$T \vdash P = \Omega\text{-}hnf(P).$$

Proof. If $T \vdash P = Q$ then, for every disjoint extension H of G, $P \sim_{\mathrm{Hrec}} Q$ follows immediately from Lemma 7.1 and Lemmas 7.5–7.11. We are thus left to show that, for every $P \in T(\Sigma \cup \Sigma_{\mathsf{FINTREE}_\Omega})$, there exists an Ω-head normal form Ω*-hnf*(P) such that

$$T \vdash P = \Omega\text{-}hnf(P). \tag{40}$$

This will follow via a straightforward structural induction on P from the following claim.

CLAIM. *Suppose* f *is an* l*-ary operation symbol from* Σ, *and* $P_1, ..., P_l \in \mathbb{T}(\Sigma_G)$ *are all in* Ω*-head normal form. Then there exists a term* $P \in \mathbb{T}(\Sigma_G)$ *in* Ω*-head normal form such that* $T \vdash f(P_1, ..., P_l) = P$.

The proof of the claim proceeds by induction on the sum of the sizes of $P_1, ..., P_l$. There are four cases to examine.

Case 1. There is an argument i that is tested positively by f and for which P_i is of the form $P_i' + P_i''$. As f is weakly distinctive, all rules for f test i positively. In this case we can apply one of the distributivity laws (35) to infer that

$$\begin{aligned} T \vdash f(P_1, ..., P_l) &= f(P_1, ..., P_i' + P_i'', ..., P_l) \\ &= f(P_1, ..., P_i', ..., P_l) \\ &\quad + f(P_1, ..., P_i'', ..., P_l). \end{aligned}$$

Next the induction hypothesis gives that there exist Ω-head normal forms P' and P'' such that $T \vdash f(P_1, ..., P_i', ..., P') = P'$ and $T \vdash f(P_1, ..., P_i'', ..., P_l) = P''$. Hence $T \vdash f(P_1, ..., P_l) = P' + P''$ and the induction step follows, after possibly using (22) once to eliminate duplicate occurrences of Ω in $P' + P''$.

Case 2. There is an argument i that is tested positively by f and for which $P_i \equiv \Omega$. Since f is weakly distinctive, all rules for f test i positively. Thus T contains a divergence law

$$f(X_1, ..., X_{i-1}, \Omega, X_{i+1}, ..., X_l) = \Omega.$$

Instantiation of this law gives $T \vdash f(\mathbf{P}) = \Omega$, and the induction step follows.

Case 3. There is an argument i that is tested positively by f and for which $P_i \equiv \delta$. We proceed by considering two subcases.

Case 3.1. There is an argument j that is tested negatively by f and for which P_j is of the form $P_j' + \Omega$. Since f is weakly distinctive, all rules for f test i positively. Thus T contains a divergence law $f(\mathbf{Q}) = \Omega$, where $Q_i \equiv \delta$, $Q_j \equiv X_j + \Omega$ and $Q_h \equiv X_h$ otherwise. Instantiation of this law gives $T \vdash f(\mathbf{P}) = \Omega$, and the induction step follows.

Case 3.2. For every argument j that is tested negatively by f, P_j does not have an Ω summand. Since f is weakly distinctive, all rules for f test i positively. Thus T contains an inaction law $T \vdash f(\mathbf{P}) = \delta$, and the induction step follows.

Case 4. For all arguments k that are tested positively by f, P_k is of the form $a_k.P_k'$. Then an application of the action law in T associated with the vector $\langle e_1, ..., e_l \rangle$, where

$$e_k = \begin{cases} a_k & \text{if } k \text{ is tested positively by } f \\ Initials(P_i) & \text{otherwise} \end{cases}$$

gives the required Ω-head normal form. ∎

We now prove that (26) holds for convergent terms built from weakly distinctive operations and the $\mathsf{FINTREE}_\Omega$ operations only, provided we add (Rec) to the equational theory given by the previous theorem.

THEOREM 7.14. *Suppose* G *is a GSOS system that disjointly extends* $\mathsf{FINTREE}_\Omega$. *Let* $\Sigma \subseteq \Sigma_G - \Sigma_{\mathsf{FINTREE}_\Omega}$ *be a collection of weakly distinctive smooth operations of* G. *Let* T *be the equational theory given by Theorem* 7.13. *Then, for every* $P \in \mathsf{CREC}(\Sigma \cup \Sigma_{\mathsf{FINTREE}_\Omega}, \mathsf{PVar})$, *if* $P{\downarrow_{Grec}}$ *then there exists a head normal form* $hnf(P)$ *such that*

$$T \cup \{(\mathrm{Rec})\} \vdash P = hnf(P).$$

Proof. By induction on the convergence predicate $\downarrow_{Grec}$ The details are very similar to those of the proof of Theorem 7.13, and are therefore omitted. We just remark here that no divergence law need be used in the proof, and that the inductive hypothesis and equation (Rec) are all that is needed to deal with the case $P \equiv \mathrm{fix}(X = Q)$, for some Q. ∎

We complete our analysis of terms built only from weakly distinctive, smooth operations and the $\mathsf{FINTREE}_\Omega$ operations by showing that the equational theory used in Theorem 7.14 is strong enough to prove property (28) for these terms.

THEOREM 7.15. *Suppose* G *is a GSOS system that disjointly extends* $\mathsf{FINTREE}_\Omega$. *Let* $\Sigma \subseteq \Sigma_G - \Sigma_{\mathsf{FINTREE}_\Omega}$ *be a collection of weakly distinctive smooth operations of* G. *Let* T *be the equational theory given by Theorem* 7.13. *Then, for every* $P \in \mathsf{CREC}(\Sigma \cup \Sigma_{\mathsf{FINTREE}_\Omega}, \mathsf{PVar})$ *and* $Q \in \mathsf{CREC}(\Sigma_G, \mathsf{PVar})$,

$$P \xrightarrow{c}_{Grec} Q \Rightarrow T \cup \{(\mathrm{Rec})\} \vdash P = P + c.Q.$$

Proof. Assume that $P \in \mathsf{CREC}(\Sigma \cup \Sigma_{\mathsf{FINTREE}_\Omega}, \mathsf{PVar})$, and that $P \xrightarrow{c}_{Grec} Q$ for some $Q \in \mathsf{CREC}(\Sigma_G, \mathsf{PVar})$. We show that $T \cup \{(\mathrm{Rec})\} \vdash P = P + c.Q$, where T is the equational theory given by Theorem 7.13. The proof of this claim is delivered in two steps, which mimic the construction of the transition relation $\rightarrow_{Grec}$ given in the proof of Proposition 4.3. First of all, we show that the claim holds when $P{\downarrow_{Grec}}$ by induction on the convergence predicate. Next we prove that the property holds in general by induction on the depth of the proof of the transition $P \xrightarrow{c}_{Grec} Q$.

Case $P{\downarrow_{Grec}}$. We show, by induction on the convergence predicate, that if $P \xrightarrow{c}_{Grec} Q$, then $T \cup \{(\mathrm{Rec})\} \vdash P = P + c.Q$. We proceed by a case analysis on the form P takes, and only consider the two non-trivial cases.

Case $P = f(\mathbf{P})$, for some $f \in \Sigma$ and vector $\mathbf{P}$ of programs in $\mathsf{CREC}(\Sigma \cup \Sigma_{\mathsf{FINTREE}_\Omega}, \mathsf{PVar})$. As the transition relation $\rightarrow_{Grec}$ is supported, the transition $P \xrightarrow{c}_{Grec} Q$ holds because there exist a rule $r \in R_G$ of the form (34), and a substitution $\sigma\colon \mathsf{Var} \to \mathsf{CREC}(\Sigma_G, \mathsf{PVar})$ such that:

1. $f(\mathbf{x})\,\sigma = f(\mathbf{P})$, i.e., $\sigma(x_i) = P_i$, for every $1 \leqslant i \leqslant l$,
2. $P_i \xrightarrow{a_i}_{Grec} \sigma(y_i)$, for every $i \in I = \mathsf{Test}^+(f)$,
3. $P_i{\downarrow_{Grec}}$ and $P_i \not\xrightarrow{b_{ij}}$ $(1 \leqslant j \leqslant n_i)$, for every $i \in K$ such that $n_i > 0$, and
4. $Q = C[\mathbf{x}, \mathbf{y}]\,\sigma$.

As P is convergent, for each argument i tested by f, $P_i{\downarrow_{Grec}}$. By the inductive hypothesis, we thus have that, for each $i \in I$,

$$T \cup \{(\mathrm{Rec})\} \vdash P_i = P_i + a_i.\sigma(y_i). \tag{41}$$

By Theorem 7.14, it follows that, for every $i \in K$ with $n_i > 0$, there exists a head normal form $hnf(P_i)$ such that

$$T \cup \{(\mathrm{Rec})\} \vdash P_i = hnf(P_i). \tag{42}$$

Consider now the substitutions $\tau, \tau'\colon \mathsf{Var} \to \mathsf{CREC}(\Sigma_G, \mathsf{PVar})$ given by

$$\tau(x) = \begin{cases} P_i + a_i.\sigma(y_i) \\ \quad \text{if} \quad x = x_i \quad \text{for some} \quad i \in I \\ hnf(P_i) \\ \quad \text{if} \quad x = x_i \quad \text{for some} \quad i \in K \quad \text{with} \quad n_i > 0 \\ \sigma(x) \\ \quad \text{otherwise} \end{cases}$$

and

$$\tau'(x) = \begin{cases} a_i.\sigma(y_i) & \text{if} \quad x = x_i \quad \text{for some} \quad i \in I \\ \tau(x) & \text{otherwise.} \end{cases}$$

Note that, as f is smooth, for no $i \in I$ does x_i occur in the context $C[\mathbf{x}, \mathbf{y}]$. Thus, for every meta-variable x occurring in $C[\mathbf{x}, \mathbf{y}]$,

$$T \cup \{(\mathrm{Rec})\} \vdash \sigma(x) = \tau'(x). \tag{43}$$

Now, by (41) and (42),

$$\begin{aligned} T \cup \{(\mathrm{Rec})\} \vdash f(\mathbf{P}) &= f(\mathbf{x})\,\sigma \\ &= f(\mathbf{x})\,\tau. \end{aligned}$$

As f is weakly distinctive, T contains a distributive law for f of the form (35) for each $i \in I$. Applying these laws repeatedly to the term $f(\mathbf{x})\,\tau$, we obtain that

$$\begin{aligned} T \cup \{(\mathrm{Rec})\} \vdash f(\mathbf{P}) &= f(\mathbf{x})\,\tau \\ &= f(\mathbf{x})\,\sigma + f(\mathbf{x})\,\tau' \\ &= f(\mathbf{P}) + f(\mathbf{x})\,\tau'. \end{aligned}$$

To prove the claim, it is thus sufficient to show that

$$T \cup \{(\mathrm{Rec})\} \vdash f(\mathbf{x})\,\tau' = f(\mathbf{x})\,\tau' + c.Q. \tag{44}$$

Consider now the vector $\langle e_1, \ldots, e_l \rangle$ over $\mathsf{Act} \cup 2^{\mathsf{Act}_\perp}$ with

$$e_i = \begin{cases} a_i & \text{if} \quad i \in I \\ Initials(hnf(P_i)) & \text{if} \quad i \in K \quad \text{and} \quad n_i > 0 \\ \varnothing & \text{otherwise.} \end{cases}$$

By construction, this vector is consistent with f, and $r \in R_G(f, \langle e_1, \ldots, e_l \rangle)$. Thus T contains the instance of law (39) associated with f and $\langle e_1, \ldots, e_l \rangle$. Using this law and Eqs. (20)–(22), we derive that

$$\begin{aligned} T \cup \{(\mathrm{Rec})\} &\vdash f(\mathbf{x})\,\tau' \\ &= f(\mathbf{x})\,\tau' + c.C[\mathbf{x}, \mathbf{y}]\,\tau' \\ &= f(\mathbf{x})\,\tau' + c.C[\mathbf{x}, \mathbf{y}]\,\sigma \quad \text{(by (43))} \\ &= f(\mathbf{x})\,\tau' + c.Q. \end{aligned}$$

The proof for this case is therefore complete.

Case $P \equiv \mathrm{fix}(X = S)$, for some $X \in \mathsf{PVar}$ and some term $S \in \mathsf{REC}(\Sigma \cup \Sigma_{\mathsf{FINTREE}_\Omega}, \mathsf{PVar})$ containing at most X free. As $P \xrightarrow{c}_{Grec} Q$ and $P{\downarrow_{Grec}}$, it must be the case that $S\{P/X\} \xrightarrow{c}_{Grec} Q$ and $S\{P/X\}{\downarrow_{Grec}}$. By the inductive hypothesis, we then have that

$$T \cup \{(\mathrm{Rec})\} \vdash S\{P/X\} = S\{P/X\} + c.Q.$$

The claim now follows immediately by law (Rec).

General Case. The proof is by induction on the depth of the proof of the transition $P \xrightarrow{c}_{Grec} Q$. The details are identical to those given above, and are therefore omitted. ▮

We now extend the above results to handle general smooth operations. This is achieved by expressing these operations as a sum of weakly distinctive operations, in very much the same way as the merge operation of ACP is expressed as a sum of the auxiliary operations of left-merge and communication merge (see, e.g., [18] for a textbook presentation). In particular, the following proposition allows for the "discovery" of, e.g., the auxiliary operations of ACP. See [7] for details and examples.

PROPOSITION 7.16. *Let G be a GSOS system that disjointly extends $\mathsf{FINTREE}_\Omega$. Assume that f is an l-ary smooth operation of G. Then there exists a disjoint extension G' of G with l-ary weakly distinctive, smooth operations $f_1, ..., f_n$ such that the following statements hold*:

1. *If G is compact, then G' is also compact*;

2. *For every disjoint extension H of G', and every vector of process variables $\mathbf{X}$ of length l,*

$$f(\mathbf{X}) \sim_{Hrec} f_1(\mathbf{X}) + \cdots + f_n(\mathbf{X}). \tag{45}$$

Proof. Assume that f is an l-ary smooth operation of G. We show how to partition the set R of rules for f in R_G into sets $R_1, ..., R_n$ in such a way that, for all $1 \leqslant i \leqslant n$, f is weakly distinctive in the GSOS system obtained from G by removing all the rules in $R - R_i$. This can be done by partitioning the set of rules for f according to the following equivalence relation:

$$r \equiv_f r' \Leftrightarrow r, r' \text{ are rules for } f \text{ that test the same arguments positively.}$$

Let $R_1, ..., R_n$ be the equivalence classes of rules for f determined by $\equiv_f$. Define $\Sigma_{G'}$ to be the signature obtained by extending Σ_G with fresh l-ary operation symbols $f_1, ..., f_n$. Next define $R_{G'}$ to be the set of rules obtained by extending R_G, for each i, with rules derived from the rules in R_i by replacing the operation symbol in the source by f_i. It is immediate to see that each operation f_i so defined is weakly distinctive, and that (45) holds. Moreover, if G is compact then G' also is, as we may assign to each new operation f_i the same weight as f. ▮

7.2. *General GSOS Operations*

In this subsection we show how to axiomatize non-smooth operations, thus lifting properties (25)–(28) to arbitrary GSOS languages with recursion.

First of all, we give a result that allows us to reduce the problem of axiomatizing arbitrary GSOS operations to that of axiomatizing smooth ones. The proof of the following proposition is an easy adaptation of those of [7, Lemma 4.13] and [4, Proposition 5.13], and we refer the reader to those references for the details.

PROPOSITION 7.17. *Suppose G is a GSOS system containing a non-smooth operation f with arity l. Then there exists a disjoint extension G' of G with a smooth operation f' with arity l' (possibly different from l), and there exist vectors $\mathbf{Z}$ of l distinct process variables, and $\mathbf{V}$ of l' variables in $\mathbf{Z}$ (possibly repeated), such that*

1. *if G is compact, then so is G'*;

2. *for every disjoint extension H of G', the equation $f(\mathbf{Z}) = f'(\mathbf{V})$ is sound with respect to $\sim_{Hrec}$, i.e.,*

$$f(\mathbf{Z}) \sim_{Hrec} f'(\mathbf{V}). \tag{46}$$

In particular, as detailed in [7], when applied to the priority operation θ, the construction given in the proof of the above proposition generates the $\triangle$ operation given by the rules (37), together with the equation

$$\theta(X) = X \triangle X.$$

For any GSOS system G, the methods presented so far allow us to generate a disjoint extension H of G and an inequational theory T with the required properties (25)–(28).

THEOREM 7.18. *Let G be a GSOS system. Then the disjoint extension H of G and inequational theory T produced by the algorithm of Fig. 2 have the following properties*:

1. *If G is compact, then so is H.*

2. *For every GSOS system H' that disjointly extends H, the inequalities in $T \cup \{(\mathrm{Rec})\}$ are sound with respect to $\lesssim_{H'rec}$.*

3. *T is Ω-head normalizing for every $P \in T(\Sigma_H)$.*

4. *T is head normalizing for every $P \in \mathsf{CREC}(\Sigma_H, \mathsf{PVar})$ such that $P{\downarrow}_{Hrec}$.*

5. *T absorbs transitions.*

Proof. Assume that G is a GSOS system. Let H and T be the GSOS system and the inequational theory generated by the algorithm in Fig. 2.

First of all, it is easy to see that adding a disjoint copy of $\mathsf{FINTREE}_\Omega$ to a compact GSOS system results in a compact GSOS system.[8] Therefore, by Proposition 7.16 and Lemma 7.17, it follows that H is compact if G was.

[8] The reader familiar with the development in [7] might recall that the semantic well-foundedness of a GSOS system (see [7, Definition 6.1]) is in general *not* preserved by adding a disjoint copy of FINTREE, and *a fortiori* of $\mathsf{FINTREE}_\Omega$, to it. The same is true in our setting. As shown by our developments in this section, however, the kind of problem highlighted in [7, Sect. 6.1] does not arise if one considers the syntatic notion of compactness in lieu of the semantic one of well-foundedness.

Input	A compact GSOS system G.
Output	A compact GSOS system H and an equational theory T with the properties mentioned in Thm. 7.18.

Step 1. If G does not disjointly extend $\mathsf{FINTREE}_\Omega$ then add to it a disjoint copy of $\mathsf{FINTREE}_\Omega$.

Step 2. For each operation f that is non-smooth, apply the construction of Lemma 7.17 to extend the system with a smoothed version of f, f'. Add all the resulting instances of law (46) to $T_{\mathsf{FINTREE}_\Omega}$.

Step 3. For each smooth, non-weakly-distinctive operation $f \notin \Sigma_{\mathsf{FINTREE}_\Omega}$ in the resulting system, apply the construction of Propn. 7.16 to generate smooth, distinctive operations $f_1, \ldots, f_n$. The system so-obtained is the H we were looking for. Add to the equational theory all the resulting instances of law (45).

Step 4. Add to the equational theory obtained in Step 3 the equations given by applying Theorem 7.13 to all the smooth, weakly distinctive operations in $\Sigma_H - \Sigma_{\mathsf{FINTREE}_\Omega}$. The result is the theory T we were looking for.

FIG. 2. The algorithm.

Next, note that, by applying instances of Eqs. (45) and (46) in T, we have that, for each $P \in \mathsf{Rec}(H, \mathsf{PVar})$, there exists a term P^* built from weakly distinctive operations and $\mathsf{FINTREE}_\Omega$ operations only such that $T \vdash P = P^*$.

Finally, Theorems 7.13–7.15 can be applied to derive that T has properties 3–5 mentioned in the statement of the theorem. ▮

We are now finally in a position to prove the promised partial completeness theorem for compact GSOS languages, whose statement we reiterate below.

THEOREM 6.15. *Let G be a compact GSOS system. Then there exist a compact GSOS system H and a set of Σ_H-inequations T such that*

- *H disjointly extends G and $\mathsf{FINTREE}_\Omega$, and*
- *for all $P \in T(\Sigma_H)$, $Q \in \mathsf{CREC}(\Sigma_H, \mathsf{PVar})$, $P \lesssim_{Hrec} Q$ iff $T \cup \{(4)\} \vdash P \leqslant Q$.*

Proof. An immediate corollary of the above theorem and of Theorem 7.2. ▮

Theorem 6.15 can be strengthened to a completeness theorem for the whole of the language $\mathsf{CREC}(\Sigma_H, \mathsf{PVar})$ with respect to $\lesssim_\omega$, at the price of adding an infinitary proof rule to the theory $T \cup \{(\mathrm{Rec})\}$. This proof rule, called *ω-rule* in [42], states that if every finite approximation of a program P is provably smaller than, or equal to, a program Q, then so is P. Formally

$$\frac{\forall n \in \mathbb{N}.\ P^n \leqslant Q}{P \leqslant Q} \tag{47}$$

where, for every $n \in \mathbb{N}$, P^n stands for the nth finite approximation of P. (See Section 5.1 for details).

THEOREM 7.19 (Completeness for $\lesssim_\omega$). *Let G be a compact GSOS system. Then the disjoint extension H of G and, the inequational theory T produced by the algorithm of Fig. 2 have the property that, for every $P, Q \in \mathsf{CREC}(\Sigma_H, \mathsf{PVar})$,*

$$P \lesssim_\omega Q \Leftrightarrow T \cup \{(\mathrm{Rec})\} \cup \{(47)\} \vdash P \leqslant Q.$$

Proof. We prove the two implications separately.

- *Soundness.* By Theorem 7.18, we know that the inequations in $T \cup \{(\mathrm{Rec})\}$ are sound with respect to $\lesssim_{Hrec}$. As $\lesssim_{Hrec} \subseteq \lesssim_\omega$, they are a fortiori also sound with respect to $\lesssim_\omega$. As H is a compact GSOS system, by Theorem 6.18 it follows that, for every $P, Q \in \mathsf{CREC}(\Sigma_H, \mathsf{PVar})$,

$$P \lesssim_\omega Q \Leftrightarrow \mathscr{D}[\![P]\!] \sqsubseteq_{\mathscr{D}} \mathscr{D}[\![Q]\!].$$

As the relations induced by a denotational semantics are guaranteed to be precongruences, $\lesssim_\omega$ is a precongruence over $\mathsf{CREC}(\Sigma_H, \mathsf{PVar})$.

We are now left to check that the proof rule (47) is sound with respect to $\lesssim_\omega$ over $\mathsf{CREC}(\Sigma_H, \mathsf{PVar})$. To this end, assume that $P, Q \in \mathsf{CREC}(\Sigma_H, \mathsf{PVar})$, and that $P^n \lesssim_\omega Q$ for every $n \in \mathbb{N}$. We prove that $P \lesssim_\omega Q$. In fact, as $\lesssim_\omega$ is algebraic by Theorem 6.17, the claim follows if we show that

$$\forall n \in \mathbb{N}\ \exists m \in \mathbb{N}\colon P^n \lesssim_\omega Q^m. \tag{48}$$

To prove (48), let P^n be a finite approximation of P. By assumption, we have that $P^n \lesssim_\omega Q$. As H is compact and P^n is a recursion-free term, by Corollary 6.5, there exists $t_{P^n} \in \mathsf{ST}(\mathsf{Act})$ such that $P^n \sim_{Hrec} t_{P^n}$. By Lemma 2.5, it follows that $P^n \lesssim_{Hrec} Q$, and therefore that $t_{P^n} \lesssim_{Hrec} Q$. As H is compact, Theorem 6.16 gives that there exists a finite

approximation Q^m of Q such that $t_{P^n} \lesssim_{Hrec} Q^m$. For such a Q^m, $P^n \lesssim_{Hrec} Q^m$. By Lemma 2.5, $P^n \lesssim_\omega Q^m$ follows.

We have thus shown that (48) holds. Hence $P \lesssim_\omega Q$.

• *Completeness.* Let $P, Q \in \mathsf{CREC}(\Sigma_H, \mathsf{PVar})$ be such that $P \lesssim_\omega Q$. We argue as follows:

$$
\begin{aligned}
P \lesssim_\omega Q &\Leftrightarrow \forall n \in \mathbb{N}\ \exists m \in \mathbb{N}\colon P^n \lesssim_\omega Q^m \\
&\qquad \text{(By Theorem 6.17)} \\
&\Leftrightarrow \forall n \in \mathbb{N}\ \exists m \in \mathbb{N}\colon P^n \lesssim_{Hrec} Q^m \\
&\qquad \text{(By Lemma 2.5)} \\
&\Leftrightarrow \forall n \in \mathbb{N}\ \exists m \in \mathbb{N}\colon T \cup \{(\mathrm{Rec})\} \vdash P^n \leqslant Q^m \\
&\qquad \text{(By Theorem 6.15)} \\
&\Rightarrow \forall n \in \mathbb{N}.\ T \cup \{(\mathrm{Rec})\} \vdash P^n \leqslant Q \\
&\qquad (Q^n \leqslant_\Omega Q \text{ implies } T \vdash Q^n \leqslant Q) \\
&\Rightarrow T \cup \{(\mathrm{Rec})\} \cup \{(47)\} \vdash P \leqslant Q.
\end{aligned}
$$

The proof of the theorem is now complete. ▮

8. CONCLUDING REMARKS

In this paper we have presented a general way of giving denotational semantics to compact GSOS languages [20, 24] in terms of the domain of synchronization trees $\mathscr{D}$ introduced by Abramsky in his seminal paper [1]. The class of compact GSOS languages consists of languages that have the structure of most standard process algebras; namely, they include a set of operations to construct finite, acyclic process graphs, and a facility for the recursive definition of behaviours. We have shown that the denotational semantics for compact GSOS languages automatically generated by our methods is guaranteed to be fully abstract with respect to the finitely observable part of the bisimulation preorder $\lesssim^{\mathrm{F}}$. The relation $\lesssim^{\mathrm{F}}$ has also been shown to coincide with $\lesssim_\omega$ for arbitrary GSOS languages, as defined by Bloom, Istrail, and Meyer in their original papers [20, 24].

As stepping stones towards the aforementioned fully abstract denotational semantics for compact GSOS languages, we have obtained several results of independent interest. In particular, we have offered a novel operational interpretation of GSOS languages in terms of labelled transition systems with divergence that relies heavily on a non-standard treatment of negative premises in GSOS rules. The outcome of our approach is, at least in our opinion, a simple operational semantics for GSOS languages in which negative premises are allowed to coexist with unguarded recursive definitions. In this set-up, the relations $\lesssim$ and $\lesssim_\omega$ are guaranteed to be precongruences for arbitrary GSOS systems, and a general algorithm, along the lines of those in [7], provides partially complete inequational axiomatizations for them in the sense of Hennessy [39]. Moreover, if the GSOS language under consideration is compact, our results guarantee that the preorder $\lesssim_\omega$ is algebraic, in the sense of, e.g., [42]; in this case, the partially complete axiomatization generated by our methods can be extended to a complete proof system over the whole of a compact GSOS language in standard fashion. The byproduct of the methods presented in this paper is a trinity of semantic views of processes (behavioural, axiomatic, and denotational) that are guaranteed to be in complete agreement.

We hope that some of these general results, whose proofs for specific process description languages tend to be quite involved and mostly use variations on the same techniques, will turn out to be useful to some of the members of our research community.

8.1. *The Benefits of Compactness*

In this paper we have chosen to present in detail three different semantic theories for compact GSOS languages. The main benefit of working with this class of GSOS languages is that the whole body of results of algebraic semantics from references like [30, 34, 38, 42] has been at our disposal. This has allowed us to present general and, we hope, rather strong results on prebisimulation, denotational models and axiomatic semantics for such languages. Moreover, the class of compact GSOS languages includes many of the standard process description languages (at least in their finite alphabet versions), and the general results that rely on the compactness of the language under consideration we have been able to establish (e.g., the algebraicity of $\lesssim_\omega$ and the algorithm for generating a partially complete axiomatization for prebisimulation) apply to these languages. However, there are indeed some process description languages considered in the literature that are *not* compact. A notable example of a non-compact process description language is SCCS with the delay operation δ [39, 63], specified by the rules

$$
\frac{}{\delta(x) \xrightarrow{1} \delta(x)} \qquad \frac{x \xrightarrow{a} x'}{\delta(x) \xrightarrow{a} x'}
$$

The presence of this operation makes SCCS non-compact, and thus some of the results that we offer in the paper do not directly apply to it. However, the delay operation is not primitive in SCCS. In fact, for any process term P, it is easy to see that the following equality holds:

$$
\delta(P) \sim \mathrm{fix}(X = 1.X + P).
$$

(The reader familiar with [39] will have noticed that Hennessy uses the above equation to deal with the δ operation in constructing the term model for SCCS given in that reference.) Therefore restricting attention to the compact fragment of SCCS, as Abramsky does in [1], allows us to

use our general results for compact GSOS languages to study properties of prebisimulation, and does not lose any expressive power. Similar considerations apply to non-compact versions of ACP obtained by adding variations on the Kleene star operation [52] to the set of core operations. (See, e.g., [19, 33].) For example, the prefix-iteration operation considered in [33] is specified by the rules

$$\frac{}{a^*x \xrightarrow{a} a^*x} \qquad \frac{x \xrightarrow{b} x'}{a^*x \xrightarrow{b} x'}$$

For any process term P, it can be checked that

$$a^*P \sim \mathrm{fix}(X = a.X + P).$$

An example of an operation which, in conjunction with the δ operation, is responsible for the non-compactness of a language and that cannot be dealt with by reduction to a suitable recursive definition (at least in the setting considered in this paper) is the desynchronizing operation Δ given by the rules (one such rule for each $a \in \mathsf{Act}$)

$$\frac{x \xrightarrow{a} x'}{\Delta(x) \xrightarrow{a} \delta(\Delta(x'))}$$

This operation is present in the early versions of SCCS studied in [39, 62, 81]. As remarked in [81, p. 249], this operation is primitive; i.e., it cannot be expressed in terms of the other operations of SCCS.

8.2. *Further Work*

We plan to direct our research effort to the study of cpo-based denotational models for arbitrary GSOS languages, in such a way that many of the results we have obtained for compact languages carry over to the more general class. As mentioned in the main body of the paper, the preorder $\lesssim_\omega$ is, in general, not going to be algebraic over arbitrary GSOS languages. As highlighted by Lemma 5.1 and the discussion which follows the proof of Theorem 6.17, this essentially depends on the fact that, for such languages, the standard syntactic notion of finite approximation from [30, 38, 42] is in some sense inappropriate, as these terms may be semantically infinite. This means that more involved techniques might be needed for proofs of full abstraction. The approach to this kind of results developed in [39] might provide important hints for the solution. We think that it would also be interesting to develop the above theory for infinitary versions of GSOS languages; i.e., GSOS languages over a denumerable set of actions, and with possibly countably infinite sets of operations and rules. The work presented in, e.g., [4, 5] should provide guidelines for restricting our attention to those infinitary GSOS languages for which a continuous semantics can be given. (Cf. [11] for an example of a language without a continuous fully abstract semantics.)

On a more speculative note, the developments in [1, 2] hint at the possibility of using the machinery developed by Abramsky in the aforementioned references and our results on denotational semantics for compact GSOS languages to automatically generate compositional proof systems for variations on Hennessy–Milner logics [44]. We think that this is a very interesting avenue for future research, but much work remains to be done in this direction.

On the operational side, it would be interesting to establish substitutivity results for other prebisimulation-like relations that have been proposed in the literature, e.g., those studied by Walker in [94] and the specification preorder of Cleaveland and Steffen [29]. In particular, the study of rule formats for bisimulation-like relations that abstract from unobservable transitions in process behaviours presented in [23] should be adapted to yield substitutivity results for (some of) the preorders studied by Walker in [94].

The work we have presented in this paper represents an attempt to generalize to a class of GSOS languages a collection of deep results that have been developed for several process description languages in the literature. We believe that this kind of meta-theoretic work is, by its own nature, experimental, and we hope that the reader will bear with us for the experimental nature of the work we offer in this study. The goodness of the results we present is the result of a trade-off between simplicity of definitions and proofs, and the degree in which our work offers the results presented in the literature when applied to particular languages.

We hope that we have given our readers convincing evidence that the theory we have developed does specialize to the known ones for, e.g., SCCS [63], CCS [65], and versions of ACP [18] with action-prefixing in lieu of general sequential composition. However, we make no claim that this work is optimal in any formal sense or applies equally well to all known languages. In fact, we believe that there is much room for improvement, but that our approach will work with minor adaptations also in the improved developments. For example, our operational interpretation of GSOS languages relies on the definition of a convergence predicate over programs. As argued in the paper, the convergence predicate given in Definition 4.1 delivers results that are in agreement with those presented in the literature for several process description languages. However, the treatment of convergence afforded by Definition 4.1 is, in certain cases, too syntactic. For example, the clauses in that definition can*not* be used to derive that the term $a; \Omega$ is convergent, where ";" stands for the sequencing operation given in [20] by the rules

$$\frac{x \xrightarrow{a} x'}{x;\, y \xrightarrow{a} x';\, y} \qquad \frac{x \not\xrightarrow{b} (\forall b \in \mathsf{Act}),\ y \xrightarrow{a} y'}{x;\, y \xrightarrow{a} y'}$$

We believe that a more general treatment of convergence can be obtained by using ruloids, in the sense of [20], in lieu of rules in an appropriate way.

We also think that there is room for improvement in the algorithm used in the proof of the partial completeness theorem for compact GSOS languages. For example, it would be nice to make the inaction laws and the action laws generated by Lemmas 7.7 and 7.11, respectively, more aesthetically pleasing.

We plan to address these issues in our future work on the topics of this paper.

ACKNOWLEDGMENTS

Many thanks to Matthew Hennessy and Edmund Robinson for their helpful remarks that led to the technical developments in Section 4. We also express our gratitude to Jan Rutten for patiently answering our questions on his work and providing pointers to the literature. The anonymous referees provided useful comments.

Received April 25, 1995; final manuscript received June 24, 1996

REFERENCES

1. Abramsky, S. (1991), A domain equation for bisimulation, *Inform. and Comput.* **92**, 161–218.
2. Abramsky, S. (1991), Domain theory in logical form, *Ann. Pure Appl. Logic.* **51**, 1–77.
3. Abramsky, S., and Vickers, S. (1993), Quantales, observational logic and process semantics, *Math. Structures Comput. Sci.* **3**, 161–227.
4. Aceto, L. (1994), "Deriving Complete Inference Systems for a Class of GSOS Languages Generating Regular Behaviours," Report IR 93-2009, Institute for Electronic Systems, Department of Mathematics and Computer Science, Aalborg University, Aalborg, November 1993. Also available as Computer Science Report 1/94, University of Sussex.
5. Aceto, L., Deriving complete inference systems for a class of GSOS languages generating regular behaviours, *in* [50, pp. 449–464].
6. Aceto, L. (1994), GSOS and finite labelled transition systems, *Theoret. Comput. Sci. Sci.* **131**, 181–195.
7. Aceto, L., Bloom, B., and Vaandrager, F. (1994), Turning SOS rules into equations, *Inform. and Comput.* **111**, 1–52.
8. Aceto, L., and Hennessy, M. (1992), Termination, deadlock and divergence, *J. Assoc. Comput. Mach.* **39**, 147/187.
9. Aceto, L., and Ingólfsdóttir, A. (1995), "CPO Models for GSOS Languages. I. Compact GSOS Languages," Research Report RS-94-40, BRICS (Basic Research in Computer Science, Centre of the Danish National Research Foundation), Department of Mathematics and Computer Science, Aalborg University, December 1994. Available by anonymous ftp at `ftp.daimi.aau.dk` in the directory `pub/BRICS 94 40`. Extended abstract *in* "Proceedings of CAAP '95," Lecture Notes in Computer Science, Vol. 915, pp. 439–453, Springer-Verlag, Berlin/New York.
10. Aczel, P. (1988), "Non-well-founded Sets," a CSLI Lecture Notes, Vol. 14, Stanford University.
11. Apt, K., and Plotkin, G. (1986), Countable nondeterminism and random assignment, *J. Assoc. Comput. Mach.* **33**, 724–767.
12. Austry, D., and Boudol, G. (1984), Algèbre de processus et synchronizations, *Theoret. Comput. Sci.* **30**, 91–131.
13. Badouel, E., and Darondeau, P. (1991), On guarded recursion, *Theoret. Comput. Sci.* **82**, 403–408.
14. Badouel, E., and Darondeau, P. (1992), Structural operational specifications and trace automata, *in* "Proceedings CONCUR 92, Stony Brook, NY" (W. Cleaveland, Ed.), Lecture Notes in Computer Science, Vol. 630, pp. 302–316, Springer-Verlag, Berlin/New York.
15. Baeten, J., and Bergstra, J. (1991), "A survey of Axiom Systems for Process Algebras," Report P9111, University of Amsterdam, Amsterdam.
16. Baeten, J., Bergstra, J., and Klop, J. (1986), Syntax and defining equations for an interrupt mechanism in process algebra, *Fund. Informat.* **9**, 127–168.
17. Beaten, J., and Verhoef, C. (1993), A congruence theorem for structured operational semantics, *in* "Proceedings CONCUR 93, Hildesheim" (E. Best, Ed.), Lecture Notes in Computer Science, Vol. 715, pp. 477–492, Springer-Verlag, Berlin/New York.
18. Baeten, J., and Weijland, W. (1990), "Process Algebra," Cambridge Tracts in Theoretical Computer Science, Vol. 18, Cambridge Univ. Press, Cambridge, UK.
19. Bergstra, J., Bethke, I., and Ponse, A. (1994), Process algebra with iteration and nesting, *Comput. J.* **37**, 243–258.
20. Bloom, B. (1989), "Ready Simulation, Bisimulation, and the Semantics of CCS-like Languages," Ph.D. thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology.
21. Bloom, B. (1991). Many meanings of monosimulation: Denotational, operational, and logical characterizations of a notion of simulation of concurrent processes. Unpublished manuscript.
22. Bloom, B. (1993), "Ready, Set, Go: Structural Operational Semantics for linear-Time Process Algebras," Tech. Rep. TR 93-1372, Department of Computer Science, Cornell University, Ithaca, New York.
23. Bloom, B. (1995), Structural operational semantics for weak bisimulations, *Theoret. Comput. Sci.* **146**, 25–68.
24. Bloom, B., Istrail, S., and Meyer, A. R. (1995), Bisimulation can't be traced, *J. Assoc. Comput. Mach.* **42**, 232–268.
25. Bol, R., and Groote, J. F. (1991), The meaning of negative premises in transition system specifications (extended abstract), *in* "Proceedings 18th - ICALP, Madrid" (J. Leach Albert, B. Monien, and M. Rodríguez, Eds.), Lecture Notes in Computer Science, Vol. 510, pp. 481–494, Springer-Verlag, Berlin/New York. Full version appeared as Report CS-R9054, CWI, Amsterdam, 1990. *Assoc. Comput. Mach.*, to appear.
26. Brookes, S., Hoare, C., and Roscoe, A. (1984), A theory of communicating sequential processes, *J. Assoc. Comput. Mach.* **31**, 560–599.
27. Brookes, S., Main, M., Melton, A., Mislove, M., and Schmidt, D. (Eds.) (1992), "Mathematical Foundations of Programming Semantics, 7th International Conference, Pittsburgh, March 1991," Lecture Notes in Computer Science, Vol. 598, Springer-Verlag, Berlin/New York.
28. Cleaveland R., and Hennessy, M. (1990), Priorities in process algebras, *Inform. and Comput.* **87**, 58–77.
29. Cleaveland, R., and Steffen, B. (1990), A preorder for partial process specification, *in* "Proceedings CONCUR 90, Amsterdam" (J. Baeten and J. Klop, Eds.), Lecture Notes in Computer Science, Vol. 458, pp. 141–151, Springer-Verlag, Berlin/New York.
30. Courcelle, B., and Nivat, M. (1976), Algebraic families of interpretations, *in* "Proceedings, 17th Symposium on Foundations of Computer Science."
31. De Nicola, R., and Hennessy, M. (1984), Testing equivalences for processes, *Theoret. Comput. Sci.* **34**, 83–133.
32. De Nicola, R., and Hennessy, M. (1987), CCS without τ's, *in* "Proceedings, TAPSOFT 87, Vol. I" (H. Ehrig, R. Kowalski, G. Levi, and U. Montanari, Eds.), Lecture Notes in Compute Science, Vol. 249, Springer-Verlag, Berlin/New York.
33. Fokkink, W. J. (1994), A complete equational axiomatization for prefix iteration, *Inform. Process. Lett.* **52**, 333–337.
34. Goguen, J., Thatcher, J., Wagner, E., and Wright, J. (1977), Initial algebra semantics and continuous algebras, *J. Assoc. Comput. Mach.* **24**, 68–95.

35. Gordon, M., Milner, R., and Wadsworth, C. (1979), "Edinburgh LCF," Lecture Notes in Computer Science, Vol. 78, Springer-Verlag, Berlin/New York.
36. Groote, J. F. (1993), Transition system specifications with negative premises, *Theoret. Comput. Sci.* **118**, 263–299.
37. Groote, J. F., and Vaandrager, F. (1992), Structured operational semantics and bisimulation as a congruence, *Inform. and Comput.* **100**, 202–260.
38. Guessarian, I. (1981), "Algebraic Semantics," Lecture Notes in Computer Science, Vol. 99, Springer-Verlag, Berlin/New York.
39. Hennessy, M. (1981), A term model for synchronous processes, *Inform. and Control* **51**, 58–75.
40. Hennessy, M. (1983), Synchronous and asynchronous experiments on processes, *Inform. and Control* **59**, 36–83.
41. Hennessy, M. (1985), Acceptance trees, *J. Assoc. Comput. Mach.* **32**, 896–628.
42. Hennessy, M. (1988), "Algebraic Theory of Processes," MIT Press, Cambridge, MAS.
43. Hennessy, M., and Ingólfsdóttir, A. (1993), A theory of communicating processes with value-passing, *Inform. and Comput.* **107**, 202–236.
44. Hennessy, M., and Milner, R. (1985), Algebraic laws for nondeterminism and concurrency, *J. Assoc. Comput. Mach.* **32**, 137–161.
45. Hennessy, M., and Plotkin, G. (1979), Full abstraction for a simple programming language, *in* "8th Symposium on Mathematical Foundations of Computer Science" (J. Bečvář, Ed.), Lecture Notes in Computer Science,) Vol. 74, pp. 108–120, Springer-Verlag, Berlin/New York.
46. Hennessy, M., and Milner, R. (1980), A term model for CCS, *in* "9th Symposium on Mathematical Foundations of Computer Science." (P. Dembiński, Ed.), Lecture Notes in Computer Science, Vol. 88, pp. 261–274, Springer-Verlag, Berlin/New York.
47. Hoare, C. (1985), "Communicating Sequential Processes," Prentice–Hall International, Englewood Cliffs, NJ.
48. Ingólfsdóttir, A. (1994), "Semantic Models for Communicating Process with Value-Passing," Ph.D. thesis, School of Cognitive and Computing Sciences, University of Sussex. Computer Science Report 8/94. Also available as Report R-94-2044, Department of Mathematics and Computer Science, Aalborg University.
49. Jifeng, H., and Hoare, C. (1993), From algebra to operational semantics, *Inform. Process. Lett.* **45**, 75–80.
50. Jonsson, B., and Parrow, J. (Eds.) (1994), "Proceedings CONCUR 94, Uppsala, Sweden," Lecture Notes in Computer Science, Vol. 836, Springer-Verlag, Berlin/New York.
51. Keller, R. (1976), Formal verification of parallel programs, *Comm. ACM* **19**, 371–384.
52. Kleene, S. (1956), Representation of events in nerve nets and finite automata, *in* "Automata Studies" (C. Shannon and J. McCarthy, Eds.), pp. 3–41, Princeton Univ. Press, Princeton, NJ.
53. Loeckx, J., and Sieber, K. (1987), "The Foundations of Program Verification," Wiley–Teubner Series in Computer Science, Wiley New York.
54. Madelaine, E., and Vergamini, D. (1991), Finiteness conditions and structural construction of automata for all process algebras, *in* "DIMACS Series in Discrete Mathematics and Theoretical Computer Science," Vol. 3, pp. 275–292.
55. Meyer, A. (1988), Semantical paradigms: Notes for an invited lecture, *in* "Proceedings 3th, Annual Symposium on Logic in Computer Science, Edinburgh," pp. 236–242, IEEE Computer Society Press.
56. Milne, G., and Milner, R. (1979), Concurrent processes and their syntax, *J. Assoc. Comput. Mach.* **26**, 302–321.
57. Milner, R. (1973), Processes: A mathematical model of computing agents, *in* "Proceedings Logic Colloquium 1973" (H. Rose and J. Shepherdson, Eds.), pp. 158–173, North-Holland, Amsterdam.
58. Milner, R. (1977), Fully abstract models of typed λ-calculi, *Theoret. Comput. Sci.* **4**, 1–22.
59. Milner, R. (1979), Flowgraphs and flow algebras, *J. Assoc. Comput. Mach.* **26**, 794–818.
60. Milner, R. (1980), "A Calculus of Communicating Systems," Lecture Notes in Computer Science, Vol. 92, Springer-Verlag, Berlin/New York.
61. Milner, R. (1981), A modal characterisation of observable machine behaviour, *in* "Proceedings CAAP 81" (G. Astesiano and C. Bohm, Eds.), Lecture Notes in Computer Science, Vol. 112, pp. 25–34, Springer-Verlag, Berlin/New York.
62. Milner, R. (1981), "On relating Synchrony and Asynchrony," Tech. Rep. CSR-75-80, Department of Computer Science, University of Edinburgh.
63. Milner, R. (1983), Calculi for synchrony and asynchrony, *Theoret. Comput. Sci.* **25**, 267–310.
64. Milner, R. (1989), "Operational and Algebraic Semantics of Concurrent Processes, Tech. Rep. ECS-LFCS-88-46, Department of Computer Science, University of Edinburgh.
65. Milner, R. (1989), "Communication and Concurrency," Prentice–Hall International, Englewood Cliffs, NJ.
66. Milner, R. (1993), Elements of interaction (Turing Award Lecture), *Comm. ACM* **36**, 78–89.
67. Milner, R., Parrow, J., and Walker, D. (1992), A calculus of mobile processes, I, II, *Inform. and Comput.* **100**, 1–77.
68. Park, D. (1981), Concurrency and automata on infinite sequences, *in* "5th GI Conference, Karlsruhe" (P. Deussen, Ed.), Lecture Notes in Computer Sciences, Vol. 104, pp. 167–183, Springer-Verlag, Berlin/New York.
69. Plotkin, G. (1976), A powerdomain construction, *SIAM J. Comput.* **5**, 452–487.
70. Plotkin, G. (1977), LCF considered as a programming language, *Theoret. Comput. Sci.* **5**, 223–256.
71. Plotkin, G. (1981), "Lecture Notes in Domain Theory," Univ. of Edinburgh.
72. Plotkin, G. (1981), "A structural Approach to Operational Semantics," Report DAIMI FN-19, Computer Science Department, Aarhus University.
73. Rosenthal, K. (1990), "Quantales and Their Applications," Research Notes in Mathematics, Pitman, London.
74. Rutten, J. (1990), Deriving denotational models for bisimulation from structured operational semantics. *in* "Proceedings, IFIP Working Conference on Programming Concepts and Methods, Sea of Galilee," (M. Broy and C. Jones. Eds.), North-Holland. Amsterdam.
75. Rutten, J., Nonwellfounded sets and programming language semantics, *in* [27, pp. 193–206.]
76. Rutten, J. (1992), Processes as terms: Non-well-founded models for bisimulation, *Math. Structures Comput. Sci.* **2**, 257–275.
77. Rutten, J., and Turi, D. (1994), Initial algebra and final coalgebra semantics for concurrency, *in* "A Decade of Concurrency," (J. de Bakker, W. de Roever, and G. Rozenberg, Eds.), Lecture Notes in Computer Science, Vol. 803, pp. 530–581, Springer-Verlag, Berlin/New York. Also available as Technical Report CS-R9409, CWI, Amsterdam.
78. Scott, D. (1971), The lattice of flow diagrams, *in* "Symposium on Semantics of Algorithmic Languages" (E. Engeler, Ed.), Lecture Notes in Mathematics, Vol. 188, pp. 311–366. Springer-Verlag, Berlin/New York.
79. Scott, D., and Strachey, C. (1971), Towards a mathematical semantics for computer languages, *in* "Proceedings of the Symposium on Computers and Automata," Microwave Research Institute Symposia Series, Vol. 21.
80. Simone, R. D. (1984), "Calculabilité et Expressivité dans l'Algebra de Processus Parallèles Meije," thèse de 3^e cycle, Univ. Paris 7.

81. Simone, R. D. (1985), Higher-level synchronising devices in Meije–SCCS, *Theoret. Comput. Sci.* **37**, 245–267.
82. Smith, S., From operational to denotational semantics, *in* [27, pp. 54–76].
83. Smyth, M. B. (1978), Power domains, *J. Comput. System Sci.* **16**, 23–35.
84. Stirling, C. (1987), Modal logics for communicating systems, *Theoret. Comput. Sci* **49**, 311–347.
85. Stirling, C. (1992), Modal and temporal logics, *in* "Handbook of Logic in Computer Science" (S. Abramsky, D. Gabbay, and T. Maibaum, Eds.), Vol. 2, pp. 477–563, Oxford Univ. Press, London.
86. Stoltenberg-Hansen, V., Lindström, I., and Griffor, E. (1994), "Mathematical Theory of Domains," Cambridge Tracts in Theoretical Computer Science, Vol. 22, Cambridge Univ. Press, Cambridge, UK.
87. Stoughton, A. (1988), "Fully Abstract Models of Programming Languages," Research Notes in Theoretical Computer Science, Pitman, London.
88. Stoughton, A. (1988), Substitution revisited, *Theoret. Comput. Sci.* **59**, 317–325.
89. Ulidowski, I. (1992), Equivalences on observable processes, *in* "Proceedings, 7th Annual Symposium on Logic in Computer Science. Santa Cruz, California," pp. 148–159, IEEE Comput. Soc. Press.
90. Vaandrager, F. (1991), On the relationship between process algebra and input/output automata (extended abstract), *in* "Proceedings 6th Annual Symposium on Logic in Computer Science, Amsterdam," pp. 387–398, IEEE Comput. Soc. Press.
91. Vaandrager, F. (1993), Expressiveness results for process algebras, *in* "Proceedings, REX Workshop on Semantics: Foundations and Applications, Beekbergen, The Netherlands, June 1992" (J. de Bakker, W. D. Roever, and G. Rosenberg, Eds.), Lecture Notes in Computer Science, Vol. 666, pp. 609–638, Springer-Verlag, Berlin/New York.
92. Verhoef, C. (1993), "A General Conservative Extension Theorem in Process algebra," Report CSN 93/38, Eindhoven University of Technology.
93. Verhoef, C., A congruence theorem for structured operational semantics with predicates and negative premises, *in* [50, pp. 433–448].
94. Walker, D. (1990), Bisimulation and divergence, *Inform. and Comput.* **85**, 202–241.